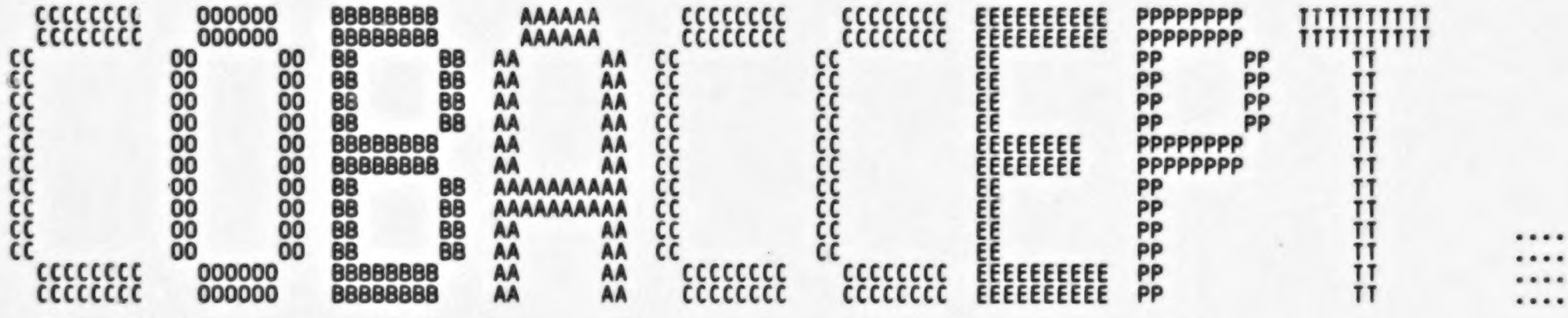

COBRA

FILEID**COBACCEPT



```
;    1          0001  0 %TITLE 'COB$ACCEPT - VAX COBOL ACCEPT Statement'
;    2          0002  0 MODULE COB$ACCEPT (                    !
;    3          0003  0                     IDENT = '1-018'    ! File: COBACCEPT.B32 EDIT:LGB1018
;    4          0004  0                                   ) =
;    5          0005  1 BEGIN
;    6          0006  1 !
;    7          0007  1 !*****************************************************************************
;    8          0008  1 !*                                                                           *
;    9          0009  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;   10          0010  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;   11          0011  1 !*  ALL RIGHTS RESERVED.                                                      *
;   12          0012  1 !*                                                                           *
;   13          0013  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;   14          0014  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;   15          0015  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;   16          0016  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;   17          0017  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;   18          0018  1 !*  TRANSFERRED.                                                              *
;   19          0019  1 !*                                                                           *
;   20          0020  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;   21          0021  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;   22          0022  1 !*  CORPORATION.                                                              *
;   23          0023  1 !*                                                                           *
;   24          0024  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;   25          0025  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;   26          0026  1 !*                                                                           *
;   27          0027  1 !*                                                                           *
;   28          0028  1 !*****************************************************************************
;   29          0029  1 !
;   30          0030  1
;   31          0031  1 !++
;   32          0032  1 ! FACILITY:  COBOL SUPPORT
;   33          0033  1 !
;   34          0034  1 ! ABSTRACT:
;   35          0035  1 !
;   36          0036  1 !       Supports the COBOL ACCEPT statement.
;   37          0037  1 !
;   38          0038  1 !       Contains COB$$OPEN_IN  to open an RMS file for input.
;   39          0039  1 !
;   40          0040  1 !
;   41          0041  1 ! ENVIRONMENT:  VAX-11 User Mode
;   42          0042  1 !
;   43          0043  1 ! AUTHOR: Rich Reichert, CREATION DATE: 16-JULY-79
;   44          0044  1 !
;   45          0045  1 ! MODIFIED BY:
;   46          0046  1 !
;   47          0047  1 ! 1-001 - Original.  RKR 16-JULY-79
;   48          0048  1 ! 1-002 - Make COB$$OPEN_IN stop instead of signal on open error.
;   49          0049  1 !         RKR 4-SEPT-79
;   50          0050  1 ! 1-003 - Make COB$$READ_RMS signal COB$_EOFON_ACC if an EOF is
;   51          0051  1 !         encountered during reading.
;   52          0052  1 !         Do string copy into caller's buffer via CH$COPY instead of
;   53          0053  1 !         STR$COPY to avoid dependency on STR$ routines.
;   54          0054  1 !         RKR 14-SEPT-79
;   55          0055  1 ! 1-004 - Identify file name on bad RMS  status other than EOF.
;   56          0056  1 !         RKR 25-SEPT-79
;   57          0057  1 ! 1-005 - Change name of symbolic LIBRARY file. RKR 1-OCT-79
```

```
;    58          0058  1 ;! 1-006 - Make module name match entry point.  RKR 20-OCT-79
;    59          0059  1 ;! 1-007 - Change references to LIB$_INVARG to COB$_INVARG.
;    60          0060  1 ;! 1-008 - Make sensitive to names in REQUIRE file.  RKR 21-OCT-79
;    61          0061  1 ;! 1-009 - Improve errors signaled.  RKR 21-OCT-79
;    62          0062  1 ;!         Cosmetic changes.  RKR 21-OCT-79
;    63          0063  1 ;! 1-010 - Imperative clean-ups, also try SYS$ logicals.
;    64          0064  1 ;!         PDG 00-FEB-81
;    65          0065  1 ;! 1-011 - Fix call to $TRNLOG to test for SS$_NORMAL
;    66          0066  1 ;!                 (since SS$_NOTRAN is a success status).
;    67          0067  1 ;!         Add COB$ACCEPT_EOF to allow ACCEPT ... AT END imp-statement.
;    68          0068  1 ;!         Allow MAX(COB$K_ACC_SIZE, .STRING[DSC$W_LENGTH]) bytes for ACCEPT.
;    69          0069  1 ;!         PDG 24-Jul-1981
;    70          0070  1 ;! 1-012 - Updated copyright date.  LB 9-Aug-81
;    71          0071  1 ;! 1-013 - Removed COB$ACCEPT_EOF.  This functionality is provided by a flag
;    72          0072  1 ;!         passed to COB$ACCEPT.
;    73          0073  1 ;! 1-014 - Add code in COB$ACCEPT to check the STV2 field in the RAB to
;    74          0074  1 ;!         determine if the terminator is an escape sequence, and if so,
;    75          0075  1 ;!         to return the escape sequence in the user's buffer.  This was
;    76          0076  1 ;!         done in response to an SPR regarding incompatibilities between
;    77          0077  1 ;!         COBOL-74 and VAX-11 COBOL.  LEB  16-FEB-82
;    78          0078  1 ;! 1-015 - Version 3 ACCEPT with screen enhancements.    LGB 15-AUG-83
;    79          0079  1 ;! 1-016 - Code converted from QIO calls to RMS.         LGB 20-JAN-84
;    80          0080  1 ;! 1-017 - Reset RAB[RAB$V_ETO] bit in COB$ACCEPT.
;    81          0081  1 ;!         Added code to COB$$ILLEGAL_TERM and COB$$DELETE_KEY to handle
;    82          0082  1 ;!         a Control Z situation.
;    83          0083  1 ;!         Added a condition to the IF statement in COB$$ILLEGAL_TERM that
;    84          0084  1 ;!         handles bell ringing for an illegal terminator.
;    85          0085  1 ;!         Put an RMS workaround in routine COB$$PARTIAL_SEQ. LGB 11-JUL-84
;    86          0086  1 ;! 1-018 - Bug fix to COB$ACCEPT - reverse IF stmt within the CH$COPY stmt.
;    87          0087  1 ;!                                                          LBG 10-SEP-84
;    88          0088  1 ;!--
```

```
;    90          0089  1 !
;    91          0090  1 ! PROLOGUE FILE
;    92          0091  1 !
;    93          0092  1 REQUIRE 'RTLIN:COBPROLOG' ;                     ! Switches, Psects, Include
;    94          1609  1                                                ! files
;    95          1610  1 !
;    96          1611  1 ! TABLE OF CONTENTS:
;    97          1612  1 !
;    98          1613  1 FORWARD ROUTINE
;    99          1614  1         COB$ACCEPT,                            ! Perform ACCEPT
;   100          1615  1         COB$ACC_SCR,                           ! Perform ACCEPT with screen enhancements
;   101          1616  1         COB$$ACC_SCR_FILE,                     ! Dealing with files not terminals,
;   102          1617  1                                                ! RMS call is different
;   103          1618  1         COB$$OPEN_IN            : NOVALUE,     ! Open for input
;   104          1619  1         COB$$RMS_GET            : NOVALUE,     ! Perform an RMS $GET call
;   105          1620  1         COB$$RMS_PUT_BYTE       : NOVALUE,     ! Perform a one byte RMS $PUT
;   106          1621  1         COB$$RMS_PUT_BUFFER     : NOVALUE,     ! Perform a buffer RMS $PUT
;   107          1622  1         COB$$CONTROL_Z          : NOVALUE,     ! Handle Control Z
;   108          1623  1         COB$$PARTIAL_SEQ        : NOVALUE,     ! Put entire escape seq in buffer
;   109          1624  1         COB$$DELETE_KEY         : NOVALUE,     ! Delete Key processing
;   110          1625  1         COB$$ILLEGAL_TERM       : NOVALUE,     ! Look for valid terminator
;   111          1626  1         COB$$CLEAN_UP           : NOVALUE,     ! Clean up before return to COBOL
;   112          1627  1         COB$$RPG_CLEAN_UP       : NOVALUE,     ! Clean up before return to RPG
;   113          1628  1         COB$$FORMAT_FOUR ;                     ! Format four CONTROL KEY routine
;   114          1629  1 !
;   115          1630  1 ! EQUATED SYMBOLS:
;   116          1631  1 !
;   117          1632  1 LITERAL
;   118          1633  1         NUM_UNITS = COB$K_UNIT_MAX - COB$K_UNIT_MIN + 1 ;
;   119          1634  1                                                ! Number of units
;   120          1635  1 LITERAL
;   121          1636  1         DISP      = 0,                         ! Code for Display
;   122          1637  1         DNA       = 1,                         ! Code for Display no Advancing
;   123          1638  1         POS       = 2,                         ! Code for Positioning
;   124          1639  1         POS_DNA   = 3,                         ! Code for Positioning no Advancing
;   125          1640  1         ACC_ADV   = 4,                         ! Code for Accept (COBOL V3)
;   126          1641  1         ACC_DNA   = 5,                         ! Code for Accept no Adv (COBOL V3)
;   127          1642  1         FLAG_MASK = 15,                        ! Mask for first four bits of
;   128          1643  1                                                ! parameter FLAGS (0-3)
;   129          1644  1         V_BELL    = 16,                        ! Bit flag for terminal bell
;   130          1645  1         V_CONV    = 32,                        ! Bit flag for conversion
;   131          1646  1         V_DEC_PT  = 64,                        ! Bit flag for 'Decimal Point
;   132          1647  1                                                ! is Comma'
;   133          1648  1         V_NO_SIGN = 128,                       ! Bit flag yes/no include sign
;   134          1649  1         V_PROTECT = 256,                       ! Bit flag for protection
;   135          1650  1         V_NO_ECHO = 512,                       ! Bit flag for no-echo
;   136          1651  1         V_ADV     = 1024,                      ! Bit flag for advancing(1)/
;   137          1652  1                                                ! no advancing (0)
;   138          1653  1         V_COB_RPG = 2048,                      ! Bit flag for VAX COBOL /
;   139          1654  1                                                ! VAX RPG
;   140          1655  1         DEL_KEY   = %X'7F',                    ! Delete key
;   141          1656  1         CZ        = %X'1A',                    ! Control Z
;   142          1657  1         CARR_RET  = 0,                         ! Parameters for routine
;   143          1658  1         LINE_FD   = 1,                         ! COB$$RMS_PUT_BYTE
;   144          1659  1         RING_BELL = 2,
;   145          1660  1         RMS_READER = 14 ;                      ! RMS uses up 14 bytes of
;   146          1661  1 !                                              ! the buffer for header info
```

```
;  147          1662  1 ! GUARDS:
;  148          1663  1 !
;  149          1664  1 !   Since the code assumes that COB$K_UNIT_MIN equals 0, and COB_TABLE
;  150          1665  1 !       has only 7 items in it, we safeguard this module.
;  151          1666  1 !
;  152          1667  1 %IF COB$K_UNIT_MIN NEQ 0 %THEN %ERROR('Unexpected COB$K_UNIT_MIN value') %FI
;  153          1668  1 %IF COB$K_UNIT_MAX GTR 6 %THEN %ERROR('Unexpected COB$K_UNIT_MAX value') %FI
;  154          1669  1
;  155          1670  1 GLOBAL
;  156          1671  1     ACC_SCR      :  INITIAL (0) ;                ! Flag for COB$ACCEPT
;  157          1672  1 OWN
;  158          1673  1     XABTRM       :  $XABTRM_DECL,                ! RMS XABTRM Control Block
;  159          1674  1     XAB_ITMLST   :  $ITMLST_DECL (ITEMS=2),      ! Item list for XABTRM
;  160          1675  1
;  161          1676  1         !+
;  162          1677  1         !   Terminator mask.
;  163          1678  1         !   Leave Control C, Y, S, and Q for VMS.
;  164          1679  1         !   Control I is Tab.  Control M is Carriage Return.
;  165          1680  1         !   ENTER key has the same value as <CR> i.e. Control M.
;  166          1681  1         !   Escape key is a terminator for a VMS V4 system.
;  167          1682  1         !   Delete key is processed via routine COB$$DELETE_KEY.
;  168          1683  1         !-
;  169          1684  1
;  170          1685  1     MASK_VECTOR :  BITVECTOR [160]              ! 20 bytes by 8 bits
;  171          1686  1                    PRESET ( [TAB] = 1,
;  172          1687  1                             [CR ] = 1,
;  173          1688  1                             [26 ] = 1,         ! Control z
;  174          1689  1                             [01 ] = 1,         ! Control a
;  175          1690  1                             [02 ] = 1,         ! Control b
;  176          1691  1                             [04 ] = 1,         ! Control d
;  177          1692  1                             [05 ] = 1,         ! Control e
;  178          1693  1                             [06 ] = 1,         ! Control f
;  179          1694  1                             [07 ] = 1,         ! Control g
;  180          1695  1                             [08 ] = 1,         ! Control h
;  181          1696  1                             [10 ] = 1,         ! Control j
;  182          1697  1                             [11 ] = 1,         ! Control k
;  183          1698  1                             [12 ] = 1,         ! Control l
;  184          1699  1                             [14 ] = 1,         ! Control n
;  185          1700  1                             [15 ] = 1,         ! Control o
;  186          1701  1                             [16 ] = 1,         ! Control p
;  187          1702  1                             [18 ] = 1,         ! Control r
;  188          1703  1                             [20 ] = 1,         ! Control t
;  189          1704  1                             [21 ] = 1,         ! Control u
;  190          1705  1                             [22 ] = 1,         ! Control v
;  191          1706  1                             [23 ] = 1,         ! Control w
;  192          1707  1                             [24 ] = 1,         ! Control x
;  193          1708  1                             [27 ] = 1,         ! Escape key for Arrow & PF Keys
;  194          1709  1                                                ! and Alternate Keypad Mode
;  195          1710  1                             [127] = 1,         ! Delete key
;  196          1711  1                             [143] = 1,         ! SS3 for Professional Keys
;  197          1712  1                             [155] = 1          ! CSI for Professional Keys
;  198          1713  1                           ) ;
;  199          1714  1 !
;  200          1715  1 !
;  201          1716  1 !   MACROS:
;  202          1717  1 !
;  203          1718  1 MACRO  COB$$B_STVO_TERM = 12,0,8,0  %;          ! Location of terminator if it
```

```
;  204          1719  1                                        ! is not an escape sequence
;  205          1720  1 MACRO  COB$$B_STV2_LEN  = 14,0,8,0  %;  ! Length of escape sequence
;  206          1721  1
;  207          1722  1 MACRO
;  208        M 1723  1     $VERIFY_TERMINATOR =
;  209        M 1724  1     !+
;  210        M 1725  1     !  If parameter KEY not sent (.KEY = 0) then CR, TAB, CONTROL Z,
;  211        M 1726  1     !  and DELETE KEY are the only legal terminators.
;  212        M 1727  1     !
;  213        M 1728  1     !  If parameter KEY not 0 then CR, TAB, CONTROL Z, DELETE KEY, PF,
;  214        M 1729  1     !  ARROW and SPECIAL FUNCTION PROFESSIONAL Keys are legal terminators.
;  215        M 1730  1     !  Copy terminator to KEY parameter if valid.
;  216        M 1731  1     !  Flag LEGAL set to 1 if terminator is valid.
;  217        M 1732  1     !-
;  218        M 1733  1           BEGIN
;  219        M 1734  1
;  220        M 1735  1           IF .TERM_SIZE EQL 1
;  221        M 1736  1           THEN
;  222        M 1737  1               BEGIN
;  223        M 1738  1               TERM_PTR = RAB [COB$$B_STV0_TERM] ;
;  224        M 1739  1               SELECTONE .RAB [COB$$B_STV0_TERM] OF
;  225        M 1740  1                   SET
;  226        M 1741  1                       [ CR,                          ! Carriage Return
;  227        M 1742  1                         TAB ] :                      ! Tab
;  228        M 1743  1
;  229        M 1744  1                           BEGIN
;  230        M 1745  1                           IF .KEY NEQ 0
;  231        M 1746  1                           THEN
;  232        M 1747  1                               CH$MOVE ( 1, .TERM_PTR, .KEY [DSC$A_POINTER] ) ;
;  233        M 1748  1                           LEGAL = 1 ;
;  234        M 1749  1                           END ;
;  235        M 1750  1
;  236        M 1751  1                       [ CZ ] :                       ! Control z
;  237        M 1752  1
;  238        M 1753  1                           BEGIN
;  239        M 1754  1                           !+
;  240        M 1755  1                           !  CONTROL Z hit along with data
;  241        M 1756  1                           !-
;  242        M 1757  1                           IF (.FLAGS AND V_COB_RPG) NEQ 0
;  243        M 1758  1                           THEN
;  244        M 1759  1                               BEGIN                  ! Control Z is illegal
;  245        M 1760  1                               LEGAL = 0 ;            ! for VAX RPG
;  246        M 1761  1                               TERM_SIZE = 0 ;
;  247        M 1762  1                               END
;  248        M 1763  1                           ELSE                       ! Special meaning for
;  249        M 1764  1                               BEGIN                  ! VAX COBOL
;  250        M 1765  1                               COB$$CLEAN_UP ( .PARAMETERS, .FLAGS ) ;
;  251        M 1766  1                               COB$$CONTROL_Z ( .UNIT, .KEY ) ;
;  252        M 1767  1                               RETURN 0 ;
;  253        M 1768  1                               END ;
;  254        M 1769  1                           END ;
;  255        M 1770  1
;  256        M 1771  1                       [ DEL_KEY ] :                  ! Delete key
;  257        M 1772  1
;  258        M 1773  1                           BEGIN
;  259        M 1774  1                           COB$$DELETE_KEY ( .PARAMETERS, .UNIT, .FLAGS ) ;
;  260        M 1775  1                           NO_BELL = 1 ;              ! Special processing for
```

```
;  261          M 1776  1                        END ;                           ! the DELETE KEY.
;  262          M 1777  1
;  263          M 1778  1                    [OTHERWISE] :                       ! Error - key not a
;  264          M 1779  1                                                        ! legal terminator
;  265          M 1780  1                        BEGIN
;  266          M 1781  1                        LEGAL = 0 ;
;  267          M 1782  1                        TERM_SIZE = 0 ;
;  268          M 1783  1                        END ;
;  269          M 1784  1                TES ;
;  270          M 1785  1            END
;  271          M 1786  1
;  272          M 1787  1        ELSE
;  273          M 1788  1            IF .CHARS_READ EQL 0  AND  .RAB [RAB$L_STS] EQL RMS$_EOF
;  274          M 1789  1            THEN
;  275          M 1790  1                !+
;  276          M 1791  1                ! CONTROL Z hit alone
;  277          M 1792  1                !-
;  278          M 1793  1                BEGIN
;  279          M 1794  1                IF (.FLAGS AND V_COB_RPG) NEQ 0
;  280          M 1795  1                THEN
;  281          M 1796  1                    BEGIN                           ! Control Z is illegal
;  282          M 1797  1                    LEGAL = 0 ;                     ! for VAX RPG
;  283          M 1798  1                    TERM_SIZE = 0 ;
;  284          M 1799  1                    END
;  285          M 1800  1                ELSE                                ! Special meaning for
;  286          M 1801  1                    BEGIN                           ! VAX COBOL
;  287          M 1802  1                    COB$$CLEAN_UP ( .PARAMETERS, .FLAGS ) ;
;  288          M 1803  1                    COB$$CONTROL_Z ( .UNIT, .KEY ) ;
;  289          M 1804  1                    RETURN 0 ;
;  290          M 1805  1                    END ;
;  291          M 1806  1                END
;  292          M 1807  1            ELSE
;  293          M 1808  1                !+
;  294          M 1809  1                ! Escape Sequence as Terminator.
;  295          M 1810  1                !-
;  296          M 1811  1                IF .KEY NEQ 0
;  297          M 1812  1                THEN
;  298          M 1813  1                    !+
;  299          M 1814  1                    ! COB$$CONTROL_KEY converts terminator sequences to COBOL
;  300          M 1815  1                    ! defined sequences and fills in KEY parameter if terminator
;  301          M 1816  1                    ! is legal.
;  302          M 1817  1                    !-
;  303          M 1818  1                    BEGIN
;  304          M 1819  1                    IF NOT ( COB$$CONTROL_KEY (TERM_PTR, .TERM_SIZE, .KEY) )
;  305          M 1820  1                    THEN
;  306          M 1821  1                        BEGIN
;  307          M 1822  1                        LEGAL = 0 ;
;  308          M 1823  1                        TERM_SIZE = 0 ;
;  309          M 1824  1                        END
;  310          M 1825  1                    ELSE
;  311          M 1826  1                        LEGAL = 1 ;
;  312          M 1827  1                    END
;  313          M 1828  1                ELSE
;  314          M 1829  1                    !+
;  315          M 1830  1                    ! KEY parameter not passed.  Escape sequences are not
;  316          M 1831  1                    ! legal terminators.
;  317          M 1832  1                    !-
```

```
;   318         M 1833  1                       BEGIN
;   319         M 1834  1                       LEGAL = 0 ;
;   320         M 1835  1                       TERM_SIZE = 0 ;
;   321         M 1836  1                       END ;
;   322         M 1837  1                 END ;                                 ! End $VERIFY_TERMINATOR macro
;   323           1838  1     % ;
;   324           1839  1
;   325           1840  1 MACRO
;   326         M 1841  1     $ERROR_REPROMPT =
;   327         M 1842  1     !+
;   328         M 1843  1     !  Ring terminal bell to signal a CONVERSION  error was been made during
;   329         M 1844  1     !  data input, restore cursor to original position and perform another
;   330         M 1845  1     !  $GET to look for valid data.
;   331         M 1846  1     !-
;   332         M 1847  1
;   333         M 1848  1     BEGIN                                             ! Begin $ERROR_REPROMPT macro
;   334         M 1849  1
;   335         M 1850  1     LOCAL
;   336         M 1851  1         PUT_TOTAL       :  INITIAL (0),               ! # of chars to $PUT
;   337         M 1852  1         INDEX           :  INITIAL (0),               ! Pointer to RESTORE_CURSOR
;   338         M 1853  1         RESTORE_CURSOR  :  VECTOR [5000, BYTE];       ! Holds sequence for restoring
;   339         M 1854  1                                                       ! cursor position.
;   340         M 1855  1     COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
;   341         M 1856  1
;   342         M 1857  1     !+
;   343         M 1858  1     !  If NO ECHO was set no need to do any erasing of input data
;   344         M 1859  1     !-
;   345         M 1860  1
;   346         M 1861  1     IF .YES_NO_ECHO EQL 0
;   347         M 1862  1     THEN
;   348         M 1863  1         BEGIN
;   349         M 1864  1
;   350         M 1865  1         IF (.PUT_FLAG NEQ 0) AND (.YES_PROTECT  EQL  0)
;   351         M 1866  1         THEN
;   352         M 1867  1
;   353         M 1868  1         !+
;   354         M 1869  1         !  If no protection set and attributes were turned on - turn them off.
;   355         M 1870  1         !  If protection was set, leave the FIELD on the screen.  OFF_BUF holds
;   356         M 1871  1         !  escape sequence to turn off attributes.  OFF_LEN - length of that
;   357         M 1872  1         !  sequence.
;   358         M 1873  1         !-
;   359         M 1874  1
;   360         M 1875  1             BEGIN
;   361         M 1876  1             CH$MOVE ( .OFF_LEN, OFF_BUF [0], RESTORE_CURSOR [0] ) ;
;   362         M 1877  1             PUT_TOTAL = .OFF_LEN ;                    ! Total for $PUT so far
;   363         M 1878  1             END ;
;   364         M 1879  1
;   365         M 1880  1         !+
;   366         M 1881  1         !  Sequence for reprompting - Backspace, Space, Backspace for each input
;   367         M 1882  1         !  character.
;   368         M 1883  1         !-
;   369         M 1884  1
;   370         M 1885  1         INDEX = .PUT_TOTAL ;                      ! PUT_TOTAL = 0 or .OFF_LEN
;   371         M 1886  1         INCR P FROM .PUT_TOTAL TO (.PUT_TOTAL+(.CHARS_READ-1)) DO
;   372         M 1887  1             BEGIN
;   373         M 1888  1             RESTORE_CURSOR [.INDEX]   = BS ;          ! Backspace
;   374         M 1889  1             RESTORE_CURSOR [.INDEX+1] = BLANK ;       ! Space
```

```
;  375          M 1890  1              RESTORE_CURSOR [.INDEX+2] = BS ;             ! Backspace
;  376          M 1891  1              INDEX = .INDEX + 3 ;
;  377          M 1892  1              END ;
;  378          M 1893  1          PUT_TOTAL = .PUT_TOTAL + (.CHARS_READ*3) ;      ! Total for $PUT so far
;  379          M 1894  1
;  380          M 1895  1          IF (.PUT_FLAG NEQ 0) AND (.YES_PROTECT  EQL  0)
;  381          M 1896  1          THEN
;  382          M 1897  1
;  383          M 1898  1          !+
;  384          M 1899  1          ! If no protection set and attributes used - turn them on again.
;  385          M 1900  1          ! (after deleting all characters from screen).  ON_BUF holds escape
;  386          M 1901  1          ! sequence to turn on attributes.  ON_LEN - length of that sequence.
;  387          M 1902  1          !-
;  388          M 1903  1
;  389          M 1904  1              BEGIN
;  390          M 1905  1              CH$MOVE ( .ON_LEN, ON_BUF [0], RESTORE_CURSOR [.PUT_TOTAL] ) ;
;  391          M 1906  1              PUT_TOTAL = .PUT_TOTAL + .ON_LEN ;          ! Total for $PUT
;  392          M 1907  1              END ;
;  393          M 1908  1
;  394          M 1909  1          END ;
;  395          M 1910  1
;  396          M 1911  1      !+
;  397          M 1912  1      ! Max for $PUT buffer is 1024 (can be increased by changing the max on
;  398          M 1913  1      ! a SYSGEN parameter).  If user input 500 characters the total sequence
;  399          M 1914  1      ! for reprompting would be 1500 bytes plus possible sequences for turning
;  400          M 1915  1      ! attributes off and on again, therefore perform a $PUT in sets of 1024
;  401          M 1916  1      ! until the whole buffer RESTORE_CURSOR has been written to terminal.
;  402          M 1917  1      !-
;  403          M 1918  1
;  404          M 1919  1      BEGIN
;  405          M 1920  1          LOCAL
;  406          M 1921  1              P_TOT,                                      ! Length of $PUT.
;  407          M 1922  1              LAST_WRITE : INITIAL (0) ;                  ! = 1 for final $PUT -
;  408          M 1923  1                                                          ! $PUT less than 1024 bytes
;  409          M 1924  1      WHILE .LAST_WRITE EQL 0 DO
;  410          M 1925  1          BEGIN
;  411          M 1926  1          IF .PUT_TOTAL GTR (COB$K_ACC_SIZE - RMS_HEADER) !  COB$K_ACC_SIZE = 1024
;  412          M 1927  1          THEN
;  413          M 1928  1              BEGIN                                       ! Need multiple $PUTs.
;  414          M 1929  1              P_TOT = COB$K_ACC_SIZE - RMS_HEADER ;       ! # to Write to screen this time.
;  415          M 1930  1              PUT_TOTAL = .PUT_TOTAL - .P_TOT ;           ! # still to Write.
;  416          M 1931  1              END
;  417          M 1932  1          ELSE
;  418          M 1933  1              BEGIN                                       ! Final $PUT
;  419          M 1934  1              P_TOT = .PUT_TOTAL ;
;  420          M 1935  1              LAST_WRITE = 1 ;
;  421          M 1936  1              END ;
;  422          M 1937  1          END;
;  423          M 1938  1
;  424          M 1939  1          !+
;  425          M 1940  1          ! Clear screen of invalid input
;  426          M 1941  1          !-
;  427          M 1942  1
;  428          M 1943  1          COB$$RMS_PUT_BUFFER ( RESTORE_CURSOR [0], .P_TOT, .FLAGS ) ;
;  429          M 1944  1      END ;
;  430          M 1945  1
;  431          M 1946  1      !+
```

```
;  432          M 1947  1      !  Perform another $GET - looking for valid input
;  433          M 1948  1      !-
;  434          M 1949  1
;  435          M 1950  1      RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
;  436          M 1951  1      COB$$RMS_GET ( .RAB, .FUNC_VAL, .ACC_SIZE, .PUT_HERE [DSC$A_POINTER] ) ;
;  437          M 1952  1
;  438          M 1953  1      REPROMPT_DONE = 1 ;                             ! Signal that REPROMPT has been
;  439          M 1954  1                                                      ! done.
;  440          M 1955  1      END ;                                           ! End of $ERROR_REPROMPT macro
;  441            1956  1    % ;
;  442            1957  1
;  443            1958  1  MACRO
;  444          M 1959  1      $BIND_PARAMETERS =
;  445          M 1960  1      !+
;  446          M 1961  1      !  Put data used by many of the subroutines in a vector of data.
;  447          M 1962  1      !  BIND all the separate names that can be used to identify the
;  448          M 1963  1      !  various elements of the vector.
;  449          M 1964  1      !-
;  450          M 1965  1      BIND
;  451          M 1966  1          PUT_HERE      = PARAMETERS [0]  :  BLOCK [,BYTE],  ! Buffer to hold input
;  452          M 1967  1          NEXT_CHAR     = PARAMETERS [3]  :  VECTOR [,BYTE], ! Buffer used for
;  453          M 1968  1                                                           ! PROTECTION check
;  454          M 1969  1          ACC_SIZE      = PARAMETERS [6]  :  WORD,           ! Length for RMS $GET
;  455          M 1970  1          CHARS_READ    = PARAMETERS [7],          ! Number of input characters
;  456          M 1971  1          FUNC_VAL      = PARAMETERS [8],          ! QIO Function Modifiers used
;  457          M 1972  1                                                   ! in the item list for RMS $GET
;  458          M 1973  1          TERM_SIZE     = PARAMETERS [9],          ! Size of terminator
;  459          M 1974  1          TERM_LOC      = PARAMETERS [10],         ! Location of terminator
;  460          M 1975  1          TERM_PTR      = PARAMETERS [11],         ! Pointer to terminator in buffer
;  461          M 1976  1          TERM_IN_NEXT  = PARAMETERS [12],         ! = 1 if terminator in NEXT_CHAR
;  462          M 1977  1          TERM_FROM_DEL = PARAMETERS [13],         ! Flag from COB$$DELETE_KEY to
;  463          M 1978  1                                                   ! COB$$ILLEGAL_TERM
;  464          M 1979  1          LEGAL         = PARAMETERS [14],         ! = 0 if illegal terminator hit
;  465          M 1980  1          YES_PROTECT   = PARAMETERS [15],         ! = 1 if PROTECTED requested
;  466          M 1981  1          YES_DEFAULT   = PARAMETERS [16],         ! = 1 if DEFAULT used as input
;  467          M 1982  1          PUT_FLAG      = PARAMETERS [17],         ! Flag for turning on attributes
;  468          M 1983  1          OFF_BUF       = PARAMETERS [18]  :  VECTOR [,BYTE], ! Holds esc seq to
;  469          M 1984  1                                                           ! turn off attributes
;  470          M 1985  1          OFF_LEN       = PARAMETERS [21] ;        ! Length of esc seq in OFF_BUF
;  471            1986  1      % ;
;  472            1987  1  !
;  473            1988  1  !   The following tables convert the UNIT number into a logical name.
;  474            1989  1  !
;  475            1990  1  MACRO
;  476            1991  1      DESC_(A) = UPLIT BYTE(%ASCIC A) - BASE %;
;  477            1992  1  BIND
;  478            1993  1      BASE = UPLIT(REP 0 OF (0)),
;  479            1994  1      COB_TABLE = UPLIT(
;  480            1995  1          DESC_('COB$INPUT'),
;  481            1996  1          DESC_('COB$OUTPUT'),
;  482            1997  1          DESC_('COB$CONSOLE'),
;  483            1998  1          DESC_('COB$CARDREADER'),
;  484            1999  1          DESC_('COB$PAPERTAPEREADER'),
;  485            2000  1          DESC_('COB$LINEPRINTER'),
;  486            2001  1          DESC_('COB$PAPERTAPEPUNCH')):   VECTOR[NUM_UNITS],
;  487            2002  1      SYS_TABLE = UPLIT(
;  488            2003  1          DESC_('SYS$INPUT'),
```

```
            DESC_('SYS$OUTPUT'),
            DESC_('SYS$ERROR'),
            DESC_('SYS$INPUT'),
            DESC_('SYS$INPUT'),
            DESC_('SYS$OUTPUT'),
            DESC_('SYS$OUTPUT')):            VECTOR[NUM_UNITS];

    !
    ! EXTERNAL REFERENCES:
    !
EXTERNAL ROUTINE

            COB$$CONTROL_KEY,                         ! Translate terminator
            COB$$ACC_CONVERT,                         ! Conversion routine
            COB$$OPEN_OUT   :  NOVALUE,               ! Open for output
            LIB$STOP : NOVALUE,                       ! Signals fatal error
            LIB$GET_VM,                               ! Get virtual memory
            LIB$FREE_VM,                              ! Free virtual memory
            STR$GET1_DX,                              ! Allocate a string
            STR$DUPL_CHAR,                            ! Duplicate character n times
            STR$FREE1_DX,                             ! Deallocate a string
            STR$COPY_R,                               ! Copy a string by ref
            COB$$SETUP_TERM_TYPE,                     ! Setup terminal type
            COB$$SET_ATTRIBUTES_ONLY ;                ! Set bold, reverse, blink,
                                                      ! underline
EXTERNAL LITERAL
            COB$_ERRDURACC,                           ! Error during DISPLAY
            COB$_FAIGET_VM,                           ! Failure to get VM
            COB$_EOFON_ACC,                           ! EOF on ACCEPT
            COB$_INVDEFVAL,                           ! DEFAULT value too large
            COB$_INVARG ;                             ! Invalid Argument(s)

EXTERNAL
            COB$$AL_WRITE_RAB      :  VECTOR,           ! Address of RAB
            COB$$AW_WRITE_IFI      :  VECTOR [,WORD],   ! Internal file identifiers
            COB$$AB_USPCODE        :  VECTOR [,BYTE],   ! Prefix and Post upspacing
            COB$$AB_PREV           :  VECTOR [,BYTE],   ! History of previous call
            COB$ACC_TERM_TYPE,                          ! Terminal type for ACCEPT
            COB$TERM_TYPE;                              ! Terminal type for DISPLAY
```

```
;  529          2043  1 %SBTTL 'COB$ACCEPT - Version 1 ACCEPT Statement'
;  530          2044  1 GLOBAL ROUTINE COB$ACCEPT (UNIT, STRING) =
;  531          2045  1
;  532          2046  1 !++
;  533          2047  1 ! FUNCTIONAL DESCRIPTION:
;  534          2048  1 !
;  535          2049  1 !       Reads a record from specified unit and delivers record to
;  536          2050  1 !       caller's string.
;  537          2051  1 !
;  538          2052  1 ! FORMAL PARAMETERS:
;  539          2053  1 !
;  540          2054  1 !       UNIT.rbu.va     Byte integer unit number designating the unit from
;  541          2055  1 !                       which the string is to be read, followed by byte
;  542          2056  1 !                       flag indicating whether routine should (false) abort,
;  543          2057  1 !                       or (true) return status on RMS$_EOF.
;  544          2058  1 !
;  545          2059  1 !       STRING.wt.ds    The address of a fixed-string descriptor to
;  546          2060  1 !                       receive the string read.
;  547          2061  1 !
;  548          2062  1 ! IMPLICIT INPUTS:
;  549          2063  1 !
;  550          2064  1 !       Status of whether the file in question is currently open.
;  551          2065  1 !
;  552          2066  1 ! IMPLICIT OUTPUTS:
;  553          2067  1 !
;  554          2068  1 !       Updated status of the file just used.
;  555          2069  1 !
;  556          2070  1 ! ROUTINE VALUE:
;  557          2071  1 !
;  558          2072  1 !       If .UNIT[1] is false:
;  559          2073  1 !           Unspecified.
;  560          2074  1 !
;  561          2075  1 !       If .UNIT[1] is true:
;  562          2076  1 !           Either true or false, indicating success or EOF, respectively.
;  563          2077  1 !
;  564          2078  1 ! SIDE EFFECTS:
;  565          2079  1 !
;  566          2080  1 !       Reads a record from the designated unit.
;  567          2081  1 !
;  568          2082  1 !--
;  569          2083  1
;  570          2084  2     BEGIN
;  571          2085  2     MAP
;  572          2086  2         UNIT:   VECTOR[,BYTE],
;  573          2087  2         STRING: REF BLOCK[8, BYTE];
;  574          2088  2
;  575          2089  2     LOCAL
;  576          2090  2         RAB:    REF $RAB_DECL,
;  577          2091  2         STATUS,
;  578          2092  2         DESCR:  BLOCK [8, BYTE],
;  579          2093  2         TERM_SIZE,
;  580          2094  2         BUFFER: VECTOR [COB$K_ACC_SIZE, BYTE];
;  581          2095  2
;  582          2096  2     IF .UNIT[0] GTRU COB$K_UNIT_MAX
;  583          2097  2     THEN
;  584          2098  2         LIB$STOP(COB$_INVARG);
;  585          2099  2
```

```
;  586          2100  2      ! If this file is not open, open it.
;  587          2101  2      ! 0 as second parameter to COB$$OPEN_IN signifies that VAX COBOL is used.
;  588          2102  2      !
;  589          2103  2      IF .COB$$AL_WRITE_RAB[.UNIT[0]] EQL 0
;  590          2104  2      THEN
;  591          2105  2          COB$$OPEN_IN(.UNIT[0], 0);
;  592          2106  2
;  593          2107  2      !  Perform a Linefeed when a Version 1 ACCEPT statement follows a
;  594          2108  2      !  Version 3 ACCEPT statement with advancing.
;  595          2109  2      !
;  596          2110  2      IF .COB$$AB_PREV [0] EQL ACC_ADV
;  597          2111  2      THEN COB$$RMS_PUT_BYTE ( LINE_FD, 0 ) ;
;  598          2112  2
;  599          2113  2      ! Read a record into our buffer or caller's buffer, whichever is larger.
;  600          2114  2      !
;  601          2115  2      RAB = .COB$$AL_WRITE_RAB[.UNIT[0]];
;  602          2116  2      IF .STRING[DSC$W_LENGTH] GTRU COB$K_ACC_SIZE
;  603          2117  2      THEN
;  604          2118  2          BEGIN
;  605          2119  2          RAB[RAB$W_USZ] = .STRING[DSC$W_LENGTH];
;  606          2120  2          RAB[RAB$L_UBF] = .STRING[DSC$A_POINTER];
;  607          2121  2          END
;  608          2122  2      ELSE
;  609          2123  2          BEGIN
;  610          2124  2          RAB[RAB$W_USZ] = COB$K_ACC_SIZE;
;  611          2125  2          RAB[RAB$L_UBF] = BUFFER;
;  612          2126  2          END;
;  613          2127  2
;  614          2128  2      !+
;  615          2129  2      !  Turn off RAB [RAB$V_ETO] just in case a 'screen enhancement ACCEPT'
;  616          2130  2      !  was performed before this one.  COB$$RMS_GET set RAB [RAB$V_ETO]
;  617          2131  2      !  to signal to RMS to expect an 'extented terminal' $GET.  Here we
;  618          2132  2      !  only want a simple $GET with no bells and whistles.  If RAB [RAB$V_ETO]
;  619          2133  2      !  is not turned off unwanted behavior will result.
;  620          2134  2      !-
;  621          2135  2
;  622          2136  2      RAB [RAB$V_ETO] = 0 ;
;  623          2137  2
;  624          2138  2      ! Read the record.
;  625          2139  2      !
;  626          2140  2      WHILE $GET(RAB = .RAB) EQL RMS$_RSA DO $WAIT(RAB = .RAB);
;  627          2141  2
;  628          2142  2
;  629          2143  2      IF NOT .RAB[RAB$L_STS]
;  630          2144  2      THEN
;  631          2145  2          LIB$STOP(
;  632          2146  2              (IF .RAB[RAB$L_STS] EQL RMS$_EOF
;  633          2147  2              THEN
;  634          2148  2                  IF .UNIT[1]
;  635          2149  2                  THEN RETURN 0
;  636          2150  2                  ELSE COB$_EOFON_ACC
;  637          2151  2              ELSE COB$_ERRDURACC),
;  638          2152  2              1, .RAB+RAB$C_BLN, .RAB[RAB$L_STS], .RAB[RAB$L_STV]);
;  639          2153  2
;  640          2154  2  !+
;  641          2155  2  ! Check if the terminator size is greater than 1.  If it is,
;  642          2156  2  ! this indicates that the terminator string is an escape sequence.
```

```
;  643          2157  2 ! Return the entire escape sequence in the user's buffer.
;  644          2158  2 !-
;  645          2159  2
;  646          2160  2     TERM_SIZE = .RAB[RAB$W_STV2];                       ! Get terminator size
;  647          2161  2
;  648          2162  2     CH$COPY( (IF .TERM_SIZE GTR 1
;  649          2163  2               THEN .RAB[RAB$W_RSZ] + .TERM_SIZE
;  650          2164  2               ELSE .RAB[RAB$W_RSZ] ),
;  651          2165  2          .RAB[RAB$L_UBF], %C' ', .STRING[DSC$W_LENGTH], .STRING[DSC$A_POINTER]);
;  652          2166  2
;  653          2167  2     !+
;  654          2168  2     ! VAX COBOL Version 1 / Version 3 interaction.
;  655          2169  2     ! Interaction with COB$ACC_SCR - Perform a Carriage Return if necessary
;  656          2170  2     ! and signal that this is an ACCEPT with advancing.
;  657          2171  2     !-
;  658          2172  2
;  659          2173  2     IF .ACC_SCR
;  660          2174  2     THEN
;  661          2175  2         COB$$RMS_PUT_BYTE ( CARR_RET, 0 ) ;
;  662          2176  2     COB$$AB_PREV[0] = ACC_ADV ;
;  663          2177  2
;  664          2178  2     RETURN 1;
;  665          2179  1     END;                                                ! End COB$ACCEPT


                                                                .TITLE  COB$ACCEPT COB$ACCEPT - VAX COBOL ACCEPT Statem
;                                                                               ent
                                                                .IDENT  \1-018\

                                                                .PSECT  _COB$DATA,NOEXE,  PIC,2

                                                00000000  00000 ACC_SCR::
                                                                .LONG   0                                               ;
                                                          00004 XABTRM: .BLKB   36
                                                          00028 XAB_ITMLST:
                                                                .BLKB   28
                                              0D  F5  FF  F6  00044 MASK_VECTOR:
                                                                .BYTE   -10, -1, -11, 13                                ;
                                                          00# 00048             .BYTE   0[11]                           ;
                                                          80  00053             .BYTE   -128                            ;
                                                          00  00054             .BYTE   0                               ;
                                                          80  00055             .BYTE   -128                            ;
                                                          00  00056             .BYTE   0                               ;
                                                          08  00057             .BYTE   8                               ;

                                                                .PSECT  _COB$CODE,NOWRT,  SHR,  PIC,2

                                                          00000 P.AAA:  .BLKB   0
                                          54  55  50  4E  49  24  42  4F  43  09  00000 P.AAC:  .ASCII  <9>\CCB$INPUT\                                ;
                                      54  55  50  54  55  4F  24  42  4F  43  0A  0000A P.AAD:  .ASCII  <10>\COB$OUTPUT\                               ;
                                  45  4C  4F  53  4E  4F  43  24  42  4F  43  0B  00015 P.AAE:  .ASCII  <11>\COB$CONSOLE\                              ;
52  45  44  41  45  52  44  52  41  43  24  42  4F  43  0E  00021 P.AAF:  .ASCII  <14>\COB$CARDREADER\                            ;
52  45  50  41  54  52  45  50  41  50  24  42  4F  43  13  00030 P.AAG:  .ASCII  <19>\COB$PAPERTAPEREADER\                       ;
                                                      52  45  44  41  45  0003F
45  54  4E  49  52  50  45  4E  49  4C  24  42  4F  43  0F  00044 P.AAH:  .ASCII  <15>\COB$LINEPRINTER\                         ;
                                                                  52  00053
50  45  50  41  54  52  45  50  41  50  24  42  4F  43  12  00054 P.AAI:  .ASCII  <18>\COB$PAPERTAPEPUNCH\                      ;
```

```
                                          48  43  4E  55  00063                                              ;
                                                          00067           .BLKB   1
00000044  00000030  00000021  00000015  0000000A  00000000 00068 P.AAB:   .LONG   0, 10, 21, 33, 48, 68, 84   ;
                                                  00000054 00080                                              ;
                 54  55  50  4E  49  24  53  59  53  09  00084 P.AAK:   .ASCII  <9>\SYS$INPUT\              ;
             54  55  50  54  55  4F  24  53  59  53  0A  0008E P.AAL:   .ASCII  <10>\SYS$OUTPUT\            ;
                 52  4F  52  52  45  24  53  59  53  09  00099 P.AAM:   .ASCII  <9>\SYS$ERROR\              ;
                 54  55  50  4E  49  24  53  59  53  09  000A3 P.AAN:   .ASCII  <9>\SYS$INPUT\              ;
                 54  55  50  4E  49  24  53  59  53  09  000AD P.AAO:   .ASCII  <9>\SYS$INPUT\              ;
             54  55  50  54  55  4F  24  53  59  53  0A  000B7 P.AAP:   .ASCII  <10>\SYS$OUTPUT\            ;
             54  55  50  54  55  4F  24  53  59  53  0A  000C2 P.AAQ:   .ASCII  <10>\SYS$OUTPUT\            ;
                                                          000CD           .BLKB   3
000000B7  000000AD  000000A3  00000099  0000008E  00000084 000D0 P.AAJ:   .LONG   132, 142, 153, 163, 173, 183, 194   ;
                                                  000000C2 000E8                                              ;

                                                                 BASE=                  P.AAA
                                                                 COB_TABLE=             P.AAB
                                                                 SYS_TABLE=             P.AAJ
                                                                          .EXTRN  COB$$CONTROL_KEY
                                                                          .EXTRN  COB$$ACC_CONVERT
                                                                          .EXTRN  COB$$OPEN_OUT, LIB$STOP
                                                                          .EXTRN  LIB$GET_VM, LIB$FREE_VM
                                                                          .EXTRN  STR$GET1_DX, STR$DUPL_CHAR
                                                                          .EXTRN  STR$FREE1_DX, STR$COPY_R
                                                                          .EXTRN  COB$$SETUP_TERM_TYPE
                                                                          .EXTRN  COB$$SET_ATTRIBUTES_ONLY
                                                                          .EXTRN  COB$_ERRDURACC, COB$_FAIGET_VM
                                                                          .EXTRN  COB$_EOFON_ACC, COB$_INVDEFVAL
                                                                          .EXTRN  COB$_INVARG, COB$$AL_WRITE_RAB
                                                                          .EXTRN  COB$$AW_WRITE_IFI
                                                                          .EXTRN  COB$$AB_USPCODE
                                                                          .EXTRN  COB$$AB_PREV, COB$ACC_TERM_TYPE
                                                                          .EXTRN  COB$TERM_TYPE, SYS$GET
                                                                          .EXTRN  SYS$WAIT

                                                    00FC 00000           .ENTRY  COB$ACCEPT, Save R2,R3,R4,R5,R6,R7        ; 2044
                                 57 00000000G   00   9E 00002           MOVAB   COB$$AB_PREV, R7                          ;
                                 56 00000000G   00   9E 00009           MOVAB   LIB$STOP, R6                              ;
                                 5E       FBF8  CE   9E 00010           MOVAB   -1032(SP), SP                             ;
                                 52         04  AC   9A 00015           MOVZBL  UNIT, R2                                  ; 2096
                                 06             52   91 00019           CMPB    R2, #6                                    ;
                                                09   1B 0001C           BLEQU   1$                                        ;
                                    00000000G   8F   DD 0001E           PUSHL   #COB$_INVARG                              ; 2098
                                 66             01   FB 00024           CALLS   #1, LIB$STOP                              ;
                                 53 00000000G0042    DE 00027 1$:       MOVAL   COB$$AL_WRITE_RAB[R2], R3                 ; 2103
                                                63   D5 0002F           TSTL    (R3)                                      ;
                                                09   12 00031           BNEQ    2$                                        ;
                                                7E   D4 00033           CLRL    -(SP)                                     ; 2105
                                                52   DD 00035           PUSHL   R2                                        ;
                         0000V   CF             02   FB 00037           CALLS   #2, COB$$OPEN_IN                          ;
                                 04             67   91 0003C 2$:       CMPB    COB$$AB_PREV, #4                          ; 2110
                                                08   12 0003F           BNEQ    3$                                        ;
                                 7E             01   7D 00041           MOVQ    #1, -(SP)                                 ; 2111
                         0000V   CF             02   FB 00044           CALLS   #2, COB$$RMS_PUT_BYTE                     ;
                                 52             63   D0 00049 3$:       MOVL    (R3), RAB                                 ; 2115
                                 53         08  AC   D0 0004C           MOVL    STRING, R3                                ; 2116
                         0400    8F             63   B1 00050           CMPW    (R3), #1024                               ;
```

```
                                        0B   1B 00055        BLEQU   4$                                        ;
                        20   A2         63   B0 00057        MOVW    (R3), 32(RAB)                             ; 2119
                        24   A2     04  A3   D0 0005B        MOVL    4(R3), 36(RAB)                            ; 2120
                                        0A   11 00060        BRB     5$                                        ; 2116
                        20   A2   0400  8F   B0 00062 4$:    MOVW    #1024, 32(RAB)                            ; 2124
                        24   A2         6E   9E 00068        MOVAB   BUFFER, 36(RAB)                           ; 2125
                        07   A2         10   8A 0006C 5$:    BICB2   #16, 7(RAB)                               ; 2136
                                        52   DD 00070 6$:    PUSHL   RAB                                       ; 2140
                 00000000G   00         01   FB 00072        CALLS   #1, SYS$GET                               ;
                 000182DA    8F         50   D1 00079        CMPL    R0, #99034                                ;
                                        0B   12 00080        BNEQ    7$                                        ;
                                        52   DD 00082        PUSHL   RAB                                       ;
                 00000000G   00         01   FB 00084        CALLS   #1, SYS$WAIT                              ;
                                        E3   11 0008B        BRB     6$                                        ;
                             2B     08  A2   E8 0008D 7$:    BLBS    8(RAB), 10$                               ; 2143
                             7E     08  A2   7D 00091        MOVQ    8(RAB), -(SP)                             ; 2152
                                    44  A2   9F 00095        PUSHAB  68(RAB)                                   ;
                                        01   DD 00098        PUSHL   #1                                        ; 2145
                 0001827A    8F     08  A2   D1 0009A        CMPL    8(RAB), #98938                            ; 2146
                                        0F   12 000A2        BNEQ    8$                                        ;
                             47     05  AC   E8 000A4        BLBS    UNIT+1, 14$                               ; 2148
                             50 00000000G 8F D0 000A8        MOVL    #COB$_EOFON_ACC, R0                       ;
                                        50   DD 000AF        PUSHL   R0                                        ;
                                        06   11 000B1        BRB     9$                                        ;
                                00000000G 8F DD 000B3 8$:    PUSHL   #COB$_ERRDURACC                           ; 2146
                             66         05   FB 000B9 9$:    CALLS   #5, LIB$STOP                              ;
                             50     0E  A2   3C 000BC 10$:   MOVZWL  14(RAB), TERM_SIZE                        ; 2160
                             01         50   D1 000C0        CMPL    TERM_SIZE, #1                             ; 2162
                                        09   15 000C3        BLEQ    11$                                       ;
                             51     22  A2   3C 000C5        MOVZWL  34(RAB), R1                               ; 2163
                             50         51   C0 000C9        ADDL2   R1, R0                                    ;
                                        04   11 000CC        BRB     12$                                       ;
                             50     22  A2   3C 000CE 11$:   MOVZWL  34(RAB), R0                               ; 2164
        63            20     24     B2  50   2C 000D2 12$:   MOVC5   R0, @36(RAB), #32, (R3), @4(R3)           ; 2165
                                    04  B3      000D8                                                          ;
                             07 00000000' EF E9 000DA        BLBC    ACC_SCR, 13$                              ; 2173
                                        7E   7C 000E1        CLRQ    -(SP)                                     ; 2175
                     0000V   CF         02   FB 000E3        CALLS   #2, COB$$RMS_PUT_BYTE                     ;
                             67         04   90 000E8 13$:   MOVB    #4, COB$$AB_PREV                          ; 2176
                             50         01   D0 000EB        MOVL    #1, R0                                    ; 2178
                                        04      000EE        RET                                               ;
                                        50   D4 000EF 14$:   CLRL    R0                                        ; 2179
                                        04      000F1        RET                                               ;

; Routine Size:  242 bytes,    Routine Base:  _COB$CODE + 00EC
```

```
;  667          2180  1 %SBTTL 'COB$ACC_SCR - ACCEPT with screen enhancements'
;  668          2181  1 GLOBAL ROUTINE COB$ACC_SCR ( UNIT          : VECTOR [2,BYTE],
;  669          2182  1                              STRING_DEST  : REF $STR$DESCRIPTOR,
;  670          2183  1                              FLAGS,
;  671          2184  1                              DEFAULT      : REF $STR$DESCRIPTOR,
;  672          2185  1                              SIZE,
;  673          2186  1                              KEY          : REF $STR$DESCRIPTOR,
;  674          2187  1                              LENGTH
;  675          2188  1                            ) =
;  676          2189  1
;  677          2190  1 !++
;  678          2191  1 ! FUNCTIONAL DESCRIPTION:
;  679          2192  1 !
;  680          2193  1 !       Performs COBOL ACCEPT statement with screen enhancements.
;  681          2194  1 !       Reads a record from a specified UNIT and deposits record in
;  682          2195  1 !       STRING_DEST.
;  683          2196  1 !       A call to COB$POS_ACCEPT is made by the VAX COBOL Compiler
;  684          2197  1 !       prior to the call to COB$ACC_SCR to set cursor position and
;  685          2198  1 !       perform any screen or line erasing.
;  686          2199  1 !
;  687          2200  1 ! CALLING SEQUENCE:
;  688          2201  1 !
;  689          2202  1 !       RETURN_STATUS.wlc.v = COB$ACC_SCR ( UNIT.rbu.va, STRING_DEST.mt.ds,
;  690          2203  1 !                                           [FLAGS.rlu.v], [DEFAULT.rt.dx],
;  691          2204  1 !                                           [SIZE.rlu.v], [KEY.wt.ds],
;  692          2205  1 !                                           [LENGTH.wlu.r] )
;  693          2206  1 ! FORMAL PARAMETERS:
;  694          2207  1 !
;  695          2208  1 !       UNIT.rbu.va     Array of two unsigned byte integers.
;  696          2209  1 !                       The first byte is the unit number designating the
;  697          2210  1 !                       device from which the string is to be read.
;  698          2211  1 !                       The second byte indicates whether the routine should
;  699          2212  1 !                       abort or return to the calling program.
;  700          2213  1 !                         Byte 2 = 0  -  routine will abort on control z
;  701          2214  1 !                                        and reprompt on conversion errors.
;  702          2215  1 !                                = 1  -  ( AT END )
;  703          2216  1 !                                        routine will return to calling program
;  704          2217  1 !                                        on control z and reprompt on conversion
;  705          2218  1 !                                        errors.
;  706          2219  1 !                                = 2  -  ( ON EXCEPTION )
;  707          2220  1 !                                        routine will return to calling program
;  708          2221  1 !                                        on control z and conversion errors.
;  709          2222  1 !
;  710          2223  1 !       STRING_DEST.mt.ds   Address of descriptor to receive the read input.
;  711          2224  1 !
;  712          2225  1 !       FLAGS.rlu.v     Screen enhancement flag;
;  713          2226  1 !
;  714          2227  1 !                               bit 0  -  bold
;  715          2228  1 !                               bit 1  -  reverse
;  716          2229  1 !                               bit 2  -  blink
;  717          2230  1 !                               bit 3  -  underline
;  718          2231  1 !                               bit 4  -  bell
;  719          2232  1 !                               bit 5  -  conversion
;  720          2233  1 !                               bit 6  -  decimal point is comma
;  721          2234  1 !                               bit 7  -  0 to allow space for sign in PROTECTED
;  722          2235  1 !                                         ACCEPT, 1 no allowance for sign
;  723          2236  1 !                               bit 8  -  protect
```

```
;  724          2237  1 !                                   bit 9  -  no-echo
;  725          2238  1 !                                   bit 10 -  0 advancing, 1 no advancing
;  726          2239  1 !                                   bit 11 -  0 for VAX COBOL, 1 for VAX RPG
;  727          2240  1 !
;  728          2241  1 !       DEFAULT.rt.dx   Default source moved to destination descriptor
;  729          2242  1 !                       (STRING_DEST) in the event of null input.
;  730          2243  1 !
;  731          2244  1 !       SIZE.rlu.v      Size of protected field.  Only applicable if the
;  732          2245  1 !                       protected flag is set.
;  733          2246  1 !
;  734          2247  1 !       KEY.wt.ds       Destination of the receiving field of the control key
;  735          2248  1 !
;  736          2249  1 !       LENGTH.wlu.r    Destination of the number of characters read
;  737          2250  1 !
;  738          2251  1 ! IMPLICIT INPUTS:
;  739          2252  1 !
;  740          2253  1 !       Status of whether the input file is currently open.
;  741          2254  1 !
;  742          2255  1 ! IMPLICIT OUTPUTS:
;  743          2256  1 !
;  744          2257  1 !       Updated status of file
;  745          2258  1 !
;  746          2259  1 ! ROUTINE VALUE:
;  747          2260  1 !
;  748          2261  1 !       If .UNIT[1] is false :  Unspecified.
;  749          2262  1 !       If .UNIT[1] is true  :  Either true or false, indicating success or
;  750          2263  1 !                               EOF, respectively.
;  751          2264  1 !
;  752          2265  1 ! SIDE EFFECTS:
;  753          2266  1 !
;  754          2267  1 !       Reads a record from a designated uint.
;  755          2268  1 !
;  756          2269  1 !--
;  757          2270  1
;  758          2271  2     BEGIN
;  759          2272  2
;  760          2273  2     LOCAL
;  761          2274  2         !+
;  762          2275  2         ! Note; other declarations are in the macro $BIND_PARAMETERS.
;  763          2276  2         !-
;  764          2277  2         RAB             :  REF $RAB_DECL,
;  765          2278  2         PUT_SIZE        :  WORD,                ! ACC_SIZE plus 5 (for escape
;  766          2279  2                                                 ! sequences)
;  767          2280  2         ON_BUF          :  VECTOR [20,BYTE],    ! Holds escape seq to turn on
;  768          2281  2                                                 ! terminal attributes
;  769          2282  2         ON_LEN          :  INITIAL (0),         ! Length of ON_BUF
;  770          2283  2         FUNC_VAL_2,                             ! QIO Function Modifiers used
;  771          2284  2                                                 ! in the item list for RMS $GET
;  772          2285  2         PROT_OK         :  INITIAL (0),         ! = 1 if no Protection errors
;  773          2286  2         CONV_OK         :  INITIAL (0),         ! = 1 if no Conversion errors
;  774          2287  2         REPROMPT_DONE   :  INITIAL (0),         ! = 1 if reprompt performed in
;  775          2288  2                                                 ! response to a Conversion error
;  776          2289  2         YES_CONV        :  INITIAL (0),         ! = 1 if Conversion requested
;  777          2290  2         YES_NO_ECHO     :  INITIAL (0),         ! = 1 if No-Echo requested
;  778          2291  2         YES_SIGN        :  INITIAL (1),         ! = 1 if no allowance for sign
;  779          2292  2                                                 ! given. NOTE - initialized to 1
;  780          2293  2         P_DATA_TYPE     :  INITIAL (0),         ! PP99 or 99PP data types
```

```
;  781          2294  2          ZEROES,                                     ! %X'0' filler
;  782          2295  2          BLANKS,                                     ! %C' ' filler
;  783          2296  2          PARAMETERS      :  VECTOR [22]              ! Buffer to hold data to be
;  784          2297  2                             INITIAL (REP 22 OF (0)) ; ! passed to subroutines
;  785          2298  2      BUILTIN
;  786          2299  2          NULLPARAMETER ;
;  787          2300  2
;  788          2301  2       LITERAL
;  789          2302  2          F_PROT_SIZE = 13,                           ! # of chars allowed for input
;  790          2303  2          D_PROT_SIZE = 22 ;                          ! when PROTECTED is requested
;  791          2304  2  !+                                                  ! for floating and double fl.
;  792          2305  2  !  Bind PARAMETERS to other names.
;  793          2306  2  !-
;  794          2307  2       $BIND_PARAMETERS ;
;  795          2308  2
;  796          2309  2  !+
;  797          2310  2  !  Fillers - used by STR$DUPL_CHAR, therefore they cannot be literals
;  798          2311  2  !-
;  799          2312  2       ZEROES = %X'0' ;
;  800          2313  2       BLANKS = %C' ' ;
;  801          2314  2
;  802          2315  2  !+
;  803          2316  2  !  Put ACCEPTed data from RMS $GET in this buffer.
;  804          2317  2  !-
;  805          2318  2
;  806          2319  2       PUT_HERE [DSC$W_LENGTH]  = 0 ;
;  807          2320  2       PUT_HERE [DSC$B_DTYPE]   = DSC$K_DTYPE_NL ;
;  808          2321  2       PUT_HERE [DSC$B_CLASS]   = DSC$K_CLASS_D ;
;  809          2322  2       PUT_HERE [DSC$A_POINTER] = 0 ;
;  810          2323  2
;  811          2324  2  !+
;  812          2325  2  !  Determine if PROTECTION has been requested.
;  813          2326  2  !  If so, set the size of the field by either the value of the SIZE parameter
;  814          2327  2  !  or the length field of the STRING_DEST descriptor.
;  815          2328  2  !  If no PROTECTION requested, use COB$K_ACC_SIZE (1024 - same as
;  816          2329  2  !  V1 Accept).
;  817          2330  2  !  Also make adjustments if both PROTECTION and CONVERSION are requested -
;  818          2331  2  !  add room for sign and a decimal point, in some cases look at DSC$B_DIGITS
;  819          2332  2  !  instead of DSC$W_LENGTH.
;  820          2333  2  !  'P' data types need special handling.
;  821          2334  2  !  Use STR$GET1_DX to allocate space for dynamic string PUT_HERE.
;  822          2335  2  !-
;  823          2336  2
;  824          2337  2       IF ( .FLAGS AND V_CONV ) NEQ 0 THEN YES_CONV = 1 ;     !  Avoid BLISS
;  825          2338  2       IF ( .FLAGS AND V_NO_SIGN ) NEQ 0 THEN YES_SIGN = 0 ;  !  optimization problems
;  826          2339  2
;  827          2340  2       IF ( .FLAGS AND V_PROTECT ) NEQ 0
;  828          2341  2       THEN
;  829          2342  3           BEGIN                                          ! Begin Protect Size
;  830          2343  3           YES_PROTECT = 1 ;
;  831          2344  3           IF .SIZE NEQ 0
;  832          2345  3           THEN ACC_SIZE = .SIZE                          ! Use SIZE
;  833          2346  3           ELSE
;  834          2347  4               BEGIN                                      ! Begin no SIZE param
;  835          2348  4                   LOCAL
;  836          2349  4                       pp99  :  initial (0) ;             ! Scale for PP99 data
;  837          2350  4                                                          ! type
```

```
;   838          2351  4
;   839          2352  4              pp99 = .string_dest [dsc$b_digits] + .string_dest [dsc$b_scale] ;
;   840          2353  4              ACC_SIZE = .STRING_DEST [DSC$W_LENGTH] ;    ! Use STRING_DEST
;   841          2354  4
;   842          2355  4 !+
;   843          2356  4 ! Special case "P" data types (each "P" specifies an assumed scaling position).
;   844          2357  4 ! NOTE:  All code pertaining to the "P" data type is in lowercase.  Since "P"
;   845          2358  4 ! data types are such an off the wall issue, leaving this code in lowercase is
;   846          2359  4 ! the best way to avoid "P" code interfering with "normal" data types.
;   847          2360  4 !-
;   848          2361  6              if ((.string_dest [dsc$b_class] eql dsc$k_class_sd )
;   849          2362  7                  and  ((.pp99 lss 0)                             ! P Picture of PP99
;   850          2363  5                  or (.string_dest[dsc$b_scale] gtr 0)))  ! P Picture of 99PP.
;   851          2364  4              then
;   852          2365  5                  begin                                       ! begin P data types
;   853          2366  5                  p_data_type = 1 ;
;   854          2367  5                  if .pp99 lss 0                              ! P Picture of PP99
;   855          2368  5                  then
;   856          2369  5                      acc_size = abs (.string_dest[dsc$b_scale])
;   857          2370  5                  else
;   858          2371  5                      acc_size = .pp99 ;
;   859          2372  5
;   860          2373  5                  if .yes_conv
;   861          2374  5                  then
;   862          2375  6                      begin
;   863          2376  6                      !+
;   864          2377  6                      ! Allow space for a decimal point for PP99 but not 99pp.
;   865          2378  6                      !-
;   866          2379  6                      if .pp99 lss 0
;   867          2380  6                      then
;   868          2381  6                          acc_size = .acc_size + 1 ;      ! decimal point for pp99
;   869          2382  6
;   870          2383  6                      !+
;   871          2384  6                      ! Because we are reading the digits and scale fields,
;   872          2385  6                      ! all numeric data types will need an extra space for
;   873          2386  6                      ! the sign - except Numeric Unsigned.
;   874          2387  6                      !-
;   875          2388  6
;   876          2389  6                      if .string_dest [dsc$b_dtype] neq dsc$k_dtype_nu
;   877          2390  6                      then
;   878          2391  6                          acc_size = .acc_size + 1 ;
;   879          2392  6                      !+
;   880          2393  6                      ! Additional check for VAX_11 COBOL COMP and COMP3
;   881          2394  6                      ! data types - if YES_SIGN = 0 then do not include
;   882          2395  6                      ! space for sign.
;   883          2396  6                      !-
;   884          2397  8                      if (((.string_dest [dsc$b_dtype] eql dsc$k_dtype_w  ) or
;   885          2398  8                           (.string_dest [dsc$b_dtype] eql dsc$k_dtype_wu ) or
;   886          2399  8                           (.string_dest [dsc$b_dtype] eql dsc$k_dtype_l  ) or
;   887          2400  8                           (.string_dest [dsc$b_dtype] eql dsc$k_dtype_lu ) or
;   888          2401  8                           (.string_dest [dsc$b_dtype] eql dsc$k_dtype_q  ) or
;   889          2402  8                           (.string_dest [dsc$b_dtype] eql dsc$k_dtype_qu ) or
;   890          2403  8                           (.string_dest [dsc$b_dtype] eql dsc$k_dtype_p ))
;   891          2404  7                                                           and .yes_sign eql 0 )
;   892          2405  6                      then
;   893          2406  6                          acc_size = .acc_size - 1 ;
;   894          2407  5                  end ;
```

```
;  895     2408  5                    end                                          ! end P data types
;  896     2409  5
;  897     2410  4                else
;  898     2411  4                    !+
;  899     2412  4                    ! Non P data type
;  900     2413  4                    !-
;  901     2414  5                    begin                                        ! Begin non P data type
;  902     2415  5                    IF .YES_CONV                                 ! Adjust ACC_SIZE
;  903     2416  5                    THEN
;  904     2417  5                        !+
;  905     2418  5                        ! Make room for overpunch sign.
;  906     2419  5                        ! Packed data type - check to see if sign should be
;  907     2420  5                        ! included.
;  908     2421  5                        !-
;  909     2422  6                        BEGIN
;  910     2423  7                        IF ((.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_NRO ) OR
;  911     2424  7                            (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_NLO ) OR
;  912     2425  8                            (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_P  AND
;  913     2426  7                                                            .YES_SIGN ))
;  914     2427  6                        THEN
;  915     2428  6                            ACC_SIZE = .ACC_SIZE + 1 ;
;  916     2429  6
;  917     2430  6                         !+
;  918     2431  6                         ! COMP - look at digits field plus one for sign, only if
;  919     2432  6                         ! conversion is requested.
;  920     2433  6                         ! VAX_COBOL always sends an SD descriptor for W, L, Q when
;  921     2434  6                         ! Conversion is used.
;  922     2435  6                         ! Check to see if sign should be included.
;  923     2436  6                         !-
;  924     2437  6
;  925     2438  7                        IF (.STRING_DEST [DSC$B_CLASS] EQL DSC$K_CLASS_SD )
;  926     2439  6                        THEN
;  927     2440  8                            IF (((.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_W  ) OR
;  928     2441  8                                 (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_WU ) OR
;  929     2442  8                                 (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_L  ) OR
;  930     2443  8                                 (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_LU ) OR
;  931     2444  8                                 (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_Q  ) OR
;  932     2445  8                                 (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_QU ))
;  933     2446  7                                                            AND .YES_SIGN )
;  934     2447  6                            THEN
;  935     2448  6                                ACC_SIZE = .STRING_DEST [DSC$B_DIGITS] + 1 ;
;  936     2449  6
;  937     2450  6                        !+
;  938     2451  6                        ! Floating pt - 13 for Floating, 22 for Double Floating.
;  939     2452  6                        !-
;  940     2453  6
;  941     2454  7                        IF (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_F )
;  942     2455  6                        THEN
;  943     2456  6                            ACC_SIZE = F_PROT_SIZE ;
;  944     2457  7                        IF (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_D )
;  945     2458  6                        THEN
;  946     2459  6                            ACC_SIZE = D_PROT_SIZE ;
;  947     2460  6
;  948     2461  6                        !+
;  949     2462  6                        ! Make room for decimal point
;  950     2463  6                        !-
;  951     2464  6
```

```
;  952          2465  7                    IF (.STRING_DEST [DSC$B_CLASS] EQL DSC$K_CLASS_SD )
;  953          2466  6                    THEN
;  954          2467  6                        ACC_SIZE = .ACC_SIZE + 1 ;
;  955          2468  5                    END ;
;  956          2469  4                end ;                                          ! End non P data type
;  957          2470  3            END ;                                              ! End no SIZE param
;  958          2471  2        END                                                    ! End Protect Size
;  959          2472  2    ELSE
;  960          2473  2        ACC_SIZE = COB$K_ACC_SIZE - RMS_HEADER ;               ! 1024 - 14 is same
;  961          2474  2                                                               ! limit as a DISPLAY
;  962          2475  2    !+
;  963          2476  2    ! Allocate enough room in PUT_HERE to hold the terminator escape sequences.
;  964          2477  2    ! Most sequences are 4 bytes or less.  (PUT_SIZE is 5 more than ACC_SIZE.)
;  965          2478  2    ! Note: PUT_SIZE used, not ACC_SIZE.
;  966          2479  2    !-
;  967          2480  2
;  968          2481  2        PUT_SIZE = .ACC_SIZE ;
;  969          2482  2        IF .ACC_SIZE LSS 920 THEN PUT_SIZE = .ACC_SIZE + 5 ;
;  970          2483  2
;  971          2484  2        IF NOT ( STR$GET1_DX ( %REF (.PUT_SIZE ), PUT_HERE ))
;  972          2485  2        THEN LIB$STOP ( COB$_ERRDURACC ) ;
;  973          2486  2
;  974          2487  2    !+
;  975          2488  2    ! Check first byte of UNIT param.
;  976          2489  2    ! If this file is not open, open it.  (Note: only first byte of UNIT is
;  977          2490  2    !                                           sent to COB$$OPEN_IN)
;  978          2491  2    !-
;  979          2492  2
;  980          2493  2        IF .UNIT[0] GTRU COB$K_UNIT_MAX
;  981          2494  2        THEN
;  982          2495  2            LIB$STOP ( COB$_INVARG ) ;
;  983          2496  2
;  984          2497  2        IF .COB$$AL_WRITE_RAB [ .UNIT[0] ] EQL 0
;  985          2498  2        THEN
;  986          2499  2            !+
;  987          2500  2            ! Second parameter tells COB$$OPEN_IN whether VAX COBOL (0)
;  988          2501  2            ! or VAX RPG (1) is the caller.
;  989          2502  2            !-
;  990          2503  2            COB$$OPEN_IN ( .UNIT[0],
;  991          2504  2                           IF ( .FLAGS AND V_COB_RPG ) NEQ 0
;  992          2505  2                           THEN 1
;  993          2506  2                           ELSE 0 ) ;
;  994          2507  2
;  995          2508  2        RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
;  996          2509  2
;  997          2510  2    !+
;  998          2511  2    ! Find out if the device is a terminal.
;  999          2512  2    !-
; 1000          2513  2
; 1001          2514  2        BEGIN
; 1002          2515  2        LOCAL
; 1003          2516  2            STATUS,
; 1004          2517  2            NAM_DSC  :  REF BLOCK [8,BYTE] ;
; 1005          2518  2
; 1006          2519  2        NAM_DSC = .RAB + RAB$C_BLN ;
; 1007          2520  2
; 1008          2521  3        IF .COB$ACC_TERM_TYPE EQL 0
```

```
: 1009          2522  3     THEN
: 1010          2523  4         IF NOT ( COB$$SETUP_TERM_TYPE ( .NAM_DSC [DSC$A_POINTER],
: 1011          2524  4                                         .NAM_DSC [DSC$W_LENGTH],
: 1012          2525  4                                         COB$ACC_TERM_TYPE ) )
: 1013          2526  4         THEN LIB$STOP ( COB$_ERRDURACC ) ;
: 1014          2527  4
: 1015          2528  4         IF .COB$ACC_TERM_TYPE EQL UNKNOWN
: 1016          2529  4         THEN
: 1017          2530  4             !+
: 1018          2531  4             ! If terminal is UNKNOWN then it can be assumed we are working
: 1019          2532  4             ! with files rather than terminals.  Pull out of this routine
: 1020          2533  4             ! and go to COB$$ACC_SCR_FILE which uses a slightly different
: 1021          2534  4             ! variation of the RMS $GET Service.
: 1022          2535  4             !-
: 1023          2536  4             BEGIN
: 1024          2537  4             STATUS = COB$$ACC_SCR_FILE ( .UNIT, .STRING_DEST, .FLAGS, .DEFAULT,
: 1025          2538  4                                          .LENGTH, .ACC_SIZE, PUT_HERE, .YES_CONV,
: 1026          2539  4                                          .YES_PROTECT, .YES_SIGN ) ;
: 1027          2540  4             !+
: 1028          2541  4             ! Free local string PUT_HERE
: 1029          2542  4             !-
: 1030          2543  5             IF NOT ( STR$FREE1_DX ( PUT_HERE ))
: 1031          2544  4             THEN LIB$STOP ( COB$_ERRDURACC ) ;
: 1032          2545  4
: 1033          2546  5             IF (NOT .STATUS)
: 1034          2547  4             THEN
: 1035          2548  4                 RETURN 0
: 1036          2549  4             ELSE
: 1037          2550  4                 RETURN 1 ;
: 1038          2551  4             END ;
: 1039          2552  3
: 1040          2553  2         END ;
: 1041          2554  2
: 1042          2555  2         !+
: 1043          2556  2         ! Flag to COB$ACCEPT that COB$ACC_SCR has been called.  COB$ACCEPT will
: 1044          2557  2         ! have to perform a Carriage Return.
: 1045          2558  2         !-
: 1046          2559  2
: 1047          2560  2         ACC_SCR = 1 ;
: 1048          2561  2
: 1049          2562  2         BEGIN                                          ! Begin $GET
: 1050          2563  2
: 1051          2564  2 !+
: 1052          2565  2 ! VAX COBOL Version 1 / Version 3 Interaction.
: 1053          2566  2 ! Advancing philosophy : <LF>   $GET   <CR>
: 1054          2567  2 ! <LF> based on previous call.
: 1055          2568  2 ! <CR> based on current ACCEPT using FLAGS bit 10.
: 1056          2569  2 ! If previous call requires advancing then perform a linefeed.  DISPLAY (DISP)
: 1057          2570  2 ! and ACCEPT (ACC_ADV) with advancing.  POS = call to module COB$POS_ERASE
: 1058          2571  2 ! remembers what previous call was, if advancing then POS, if no advancing
: 1059          2572  2 ! then POS_DNA
: 1060          2573  2 !-
: 1061          2574  2
: 1062          2575  4         IF (.COB$$AB_PREV[0] EQL DISP
: 1063          2576  4             OR  .COB$$AB_PREV[0] EQL POS
: 1064          2577  4             OR  .COB$$AB_PREV[0] EQL ACC_ADV )
: 1065          2578  3         THEN
```

```
;  1066          2579  3          !+
;  1067          2580  3          ! Echo linefeed to terminal
;  1068          2581  3          !-
;  1069          2582  3          COB$$RMS_PUT_BYTE ( LINE_FD, .FLAGS ) ;
;  1070          2583  3
;  1071          2584  2      !+
;  1072          2585  2      ! Did user request any terminal attributes (bold, blink, underline, reverse) ?
;  1073          2586  2      ! If so, call COB$$SET_ATTRIBUTES_ONLY to get escape sequence to turn
;  1074          2587  2      ! attributes on and off.
;  1075          2588  2      ! PUT_FLAG - first four bits (0-3) of FLAGS parameter.
;  1076          2589  2      !-
;  1077          2590  2
;  1078          2591  2          PUT_FLAG = .FLAGS  AND  FLAG_MASK ;
;  1079          2592  2
;  1080          2593  2          IF .PUT_FLAG NEQ 0
;  1081          2594  2          THEN
;  1082          2595  2              IF NOT ( COB$$SET_ATTRIBUTES_ONLY ( .COB$ACC_TERM_TYPE, .PUT_FLAG,
;  1083          2596  2                                                  ON_BUF [0], ON_LEN,
;  1084          2597  2                                                  OFF_BUF [0], OFF_LEN ) )
;  1085          2598  2              THEN LIB$STOP ( COB$_ERRDURACC ) ;
;  1086          2599  2
;  1087          2600  2      !+
;  1088          2601  2      ! If requested, add sequence to ON_BUF to ring terminal bell.
;  1089          2602  2      !-
;  1090          2603  2
;  1091          2604  2          IF ( .FLAGS AND V_BELL ) NEQ 0
;  1092          2605  2          THEN
;  1093          2606  2              BEGIN
;  1094          2607  2              ON_BUF [ .ON_LEN ] = BELL ;
;  1095          2608  2              ON_LEN = .ON_LEN + 1 ;
;  1096          2609  2              END ;
;  1097          2610  2
;  1098          2611  3      !+
;  1099          2612  3      ! Check parameters to see if the CONTROL KEY FORMAT 4 ACCEPT has been
;  1100          2613  3      ! requested.  If so, pull out of this routine and call COB$$FORMAT_FOUR
;  1101          2614  3      ! which uses a different Terminator Mask and does not need all the
;  1102          2615  3      ! enhancements in COB$ACC_SCR.
;  1103          2616  3      !-
;  1104          2617  3
;  1105          2618  3          IF NOT NULLPARAMETER (KEY)
;  1106          2619  3          THEN
;  1107          2620  4              BEGIN
;  1108          2621  4                  LOCAL
;  1109          2622  4                      KEY_LEN ;
;  1110          2623  4
;  1111          2624  4              KEY_LEN = .KEY [DSC$W_LENGTH] ;
;  1112          2625  4              STR$DUPL_CHAR ( .KEY, KEY_LEN, BLANKS ) ;
;  1113          2626  4
;  1114          2627  4              !+
;  1115          2628  4              ! If these parameters are not present then we are dealing with
;  1116          2629  4              ! a Format Four ACCEPT rather than a Format Three ACCEPT.
;  1117          2630  4              !-
;  1118          2631  4
;  1119          2632  5              IF (NULLPARAMETER (LENGTH)  AND
;  1120          2633  5                  NULLPARAMETER (SIZE)    AND
;  1121          2634  5                  NULLPARAMETER (DEFAULT) AND
;  1122          2635  5                  NULLPARAMETER (STRING_DEST) )
```

```
;  1123     2636  4                THEN
;  1124     2637  5                    IF NOT ( COB$$FORMAT_FOUR ( .UNIT, .FLAGS, .KEY ))
;  1125     2638  4                    THEN RETURN 0
;  1126     2639  4                    ELSE RETURN 1 ;
;  1127     2640  3                END ;
;  1128     2641  3
;  1129     2642  3            !+
;  1130     2643  3            !  Determine FUNC_VAL - QIO Function Modifiers used by RMS $GET Service.
;  1131     2644  3            !  Check FLAGS parameter to see if NO-ECHO was requested (bit 9), if so
;  1132     2645  3            !  set TRM$M_TM_NOECHO to suppress echoing of input characters to the terminal.
;  1133     2646  3            !  Set TRM$M_TM_ESCAPE to allow Escape Sequences to act as terminators (Arrow
;  1134     2647  3            !  keys, PF keys, and the Professional editing and top row function keys).
;  1135     2648  3            !  Set TRM$M_TM_NOFILTR to have the DELETE KEY handled by COB$$DELETE_KEY.
;  1136     2649  3            !  Set TRM$M_TM_TRMNOECHO to suppress echoing of the termination character
;  1137     2650  3            !  (COB$$AB_PREV handles advancing / no advancing).
;  1138     2651  3            !-
;  1139     2652  3
;  1140     2653  3                IF ( .FLAGS AND V_NO_ECHO ) NEQ 0
;  1141     2654  3                THEN
;  1142     2655  4                    BEGIN
;  1143     2656  4                    FUNC_VAL = TRM$M_TM_ESCAPE + TRM$M_TM_NOFILTR + TRM$M_TM_TRMNOECHO
;  1144     2657  4                                                                  + TRM$M_TM_NOECHO ;
;  1145     2658  4                    YES_NO_ECHO = 1 ;
;  1146     2659  4                    END
;  1147     2660  3                ELSE
;  1148     2661  3                    FUNC_VAL = TRM$M_TM_ESCAPE + TRM$M_TM_NOFILTR + TRM$M_TM_TRMNOECHO ;
;  1149     2662  3
;  1150     2663  3            !+
;  1151     2664  3            !  Main Loop of routine.
;  1152     2665  3            !  PROT_OK = 1 -> there was no Protection error  "plus"
;  1153     2666  3            !  CONV_OK = 1 -> there was no Conversion error  "equal"  SUCCESS -> pull out
;  1154     2667  3            !  of loop.  Otherwise continue accepting data until there are no errors.
;  1155     2668  3            !  If error, reprompt user for more input via macro $ERROR_REPROMPT.
;  1156     2669  3            !-
;  1157     2670  3
;  1158     2671  3                WHILE (.PROT_OK EQL 0)  OR  (.CONV_OK EQL 0) DO
;  1159     2672  4                    BEGIN                                       ! Begin loop
;  1160     2673  4                        LOCAL
;  1161     2674  4                            TERM_SEEN  :  INITIAL (0) ;         ! Flag for PROTECT check
;  1162     2675  4
;  1163     2676  4                    IF .REPROMPT_DONE EQL 0
;  1164     2677  4                    THEN
;  1165     2678  5                        BEGIN                                   ! Begin no reprompt
;  1166     2679  5
;  1167     2680  5                        !+
;  1168     2681  5                        !  If PROTECTION requested, put a Protected Field on the screen.
;  1169     2682  5                        !  $PUT ACC_SIZE blanks to screen with attributes requested
;  1170     2683  5                        !  by user turned on.  (Escape sequences geared to VT100
;  1171     2684  5                        !  terminals)  Can only set a one line field as a max, therefore
;  1172     2685  5                        !  FIELD_BUF holds up to 300 characters.
;  1173     2686  5                        !-
;  1174     2687  5
;  1175     2688  5                        IF .COBSACC_TERM_TYPE EQL VT100
;  1176     2689  5                        THEN
;  1177     2690  6                            BEGIN                               ! Begin VT100
;  1178     2691  6                            IF .YES_PROTECT
;  1179     2692  6                            THEN
```

```
;  1180          2693  7                     BEGIN                                       ! Begin Field
;  1181          2694  7                     LOCAL
;  1182          2695  7                         FIELD_BUF       :  VECTOR [300, BYTE],
;  1183          2696  7                         FIELD_LEN ;                             ! size of FIELD_BUF
;  1184          2697  7
;  1185          2698  7                     !+
;  1186          2699  7                     !  Buffer FIELD_BUF to write Protected Field contains
;  1187          2700  7                     !            - escape sequence to turn attributes on,
;  1188          2701  7                     !            - number of blanks to write to screen and
;  1189          2702  7                     !            - backspaces (same # as blanks) to put cursor
;  1190          2703  7                     !              back to original position.
;  1191          2704  7                     !-
;  1192          2705  7
;  1193          2706  7                     CH$MOVE ( .ON_LEN, ON_BUF [0], FIELD_BUF [0] ) ;
;  1194          2707  7                     FIELD_LEN = .ON_LEN ;
;  1195          2708  7                     CH$FILL ( BLANK, .ACC_SIZE, FIELD_BUF [.FIELD_LEN] ) ;
;  1196          2709  7                     FIELD_LEN = .FIELD_LEN + .ACC_SIZE ;
;  1197          2710  7                     CH$FILL ( BS, .ACC_SIZE, FIELD_BUF [.FIELD_LEN] ) ;
;  1198          2711  7                     FIELD_LEN = .FIELD_LEN + .ACC_SIZE ;
;  1199          2712  7
;  1200          2713  7                     !+
;  1201          2714  7                     !  If size of FIELD_BUF is greater than the size of the
;  1202          2715  7                     !  maximum allowed for a $PUT buffer, issue an error
;  1203          2716  7                     !  message.  Issuing multiple $PUTs at this point does
;  1204          2717  7                     !  not help as the cursor is unable to get back to the
;  1205          2718  7                     !  starting position and ends up in the wrong line.
;  1206          2719  7                     !-
;  1207          2720  7
;  1208          2721  8                     IF .FIELD_LEN GTR (COB$K_ACC_SIZE - RMS_HEADER) ! 1024 -14
;  1209          2722  7                     THEN
;  1210          2723  7                         LIB$STOP ( COB$_ERRDURACC ) ;
;  1211          2724  7
;  1212          2725  7                     !+
;  1213          2726  7                     !  $PUT to write Protected Field to terminal
;  1214          2727  7                     !-
;  1215          2728  7
;  1216          2729  7                     COB$$RMS_PUT_BUFFER ( FIELD_BUF [0], .FIELD_LEN, .FLAGS ) ;
;  1217          2730  7
;  1218          2731  6                     END ;                                       ! End Field
;  1219          2732  5                 END ;                                           ! End VT100
;  1220          2733  5
;  1221          2734  5 !*****
;  1222          2735  5 !*****  RMS $GET Service
;  1223          2736  5 !*****
;  1224          2737  5
;  1225          2738  5                 !+
;  1226          2739  5                 !  RMS $PUT to turn on terminal attributes (blink,bold,underline,reverse).
;  1227          2740  5                 !  RMS $GET to accept input.  Do not perform the $PUT if PROTECTED
;  1228          2741  5                 !  is requested as the FIELD_BUF $PUT has already turned attributes
;  1229          2742  5                 !  on.
;  1230          2743  5                 !  Note :  TRM$_PROMPT not used because of buffer size limitations.
;  1231          2744  5                 !    TRM$_MODIFIERS uses all of specified buffer for accepting input,
;  1232          2745  5                 !    TRM$_PROMPT uses same buffer for both the prompt string and the
;  1233          2746  5                 !    accepted data, therefore some space for accepting data is lost.
;  1234          2747  5                 !-
;  1235          2748  5
;  1236          2749  5                 IF .ON_LEN NEQ 0 AND .YES_PROTECT NEQ 1     ! If requested, turn
```

```
;  1237     2750  5              THEN                                                ! attributes on
;  1238     2751  5                  COB$$RMS_PUT_BUFFER ( ON_BUF [0], .ON_LEN, .FLAGS ) ;
;  1239     2752  5
;  1240     2753  5              !+
;  1241     2754  5              ! RMS $GET to accept input from terminal.
;  1242     2755  5              !-
;  1243     2756  5
;  1244     2757  5              RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
;  1245     2758  5              COB$$RMS_GET ( .RAB, .FUNC_VAL, .ACC_SIZE,
;  1246     2759  5                                              .PUT_HERE [DSC$A_POINTER] ) ;
;  1247     2760  5
;  1248     2761  5              END                                                 ! End of no reprompt
;  1249     2762  4          ELSE
;  1250     2763  4              REPROMPT_DONE = 0 ;                                 ! re-set flag
;  1251     2764  4
;  1252     2765  4          !+
;  1253     2766  4          ! Get number of characters read and terminator size from the fields
;  1254     2767  4          ! of the RAB.  Pass this info along to other routines.
;  1255     2768  4          ! RAB fields -
;  1256     2769  4          !       rab [rab$l_sts]        = status
;  1257     2770  4          !       rab [rab$l_rsz]        = x      no. of chars read
;  1258     2771  4          !       rab [cob$$b_stv0_term] = d      <cr> terminator seen
;  1259     2772  4          !       rab [cob$$b_stv2_len]  = 1      size of terminator
;  1260     2773  4          ! Save this information before COB$$PARTIAL_SEQ does any more $GETs.
;  1261     2774  4          !-
;  1262     2775  4
;  1263     2776  4          CHARS_READ = .RAB [RAB$W_RSZ] ;                         ! Number of chars read
;  1264     2777  4          TERM_SIZE  = .RAB [COB$$B_STV2_LEN] ;                   ! Size and location of
;  1265     2778  4          TERM_LOC   = .RAB [COB$$B_STV0_TERM] ;                  ! terminator - other
;  1266     2779  4                                                                  ! routines may update
;  1267     2780  4                                                                  ! these
;  1268     2781  4          !+
;  1269     2782  4          ! Check for partial sequence error - not enough room in input buffer
;  1270     2783  4          ! to hold entire escape sequence when a Protected ACCEPT is performed.
;  1271     2784  4          ! If necessary, call COB$$PARTIAL_SEQ to read remainder of sequence.
;  1272     2785  4          !-
;  1273     2786  4
;  1274     2787  4          IF .RAB [RAB$L_STS] EQL RMS$_PES
;  1275     2788  4          THEN
;  1276     2789  5              BEGIN
;  1277     2790  5              TERM_SEEN = 1 ;                                     ! Set flag here as
;  1278     2791  5              COB$$PARTIAL_SEQ ( PARAMETERS, .UNIT ) ;            ! COB$$PARTIAL_SEQ may
;  1279     2792  4              END ;                                               ! change status value
;  1280     2793  4
;  1281     2794  4          !+
;  1282     2795  4          ! If terminator was the DELETE KEY call COB$$DELETE_KEY.
;  1283     2796  4          !-
;  1284     2797  4
;  1285     2798  4          IF .RAB [COB$$B_STV0_TERM] EQL DEL_KEY
;  1286     2799  4          THEN
;  1287     2800  4              COB$$DELETE_KEY ( PARAMETERS, .UNIT, .FLAGS ) ;
;  1288     2801  4
;  1289     2802  4 !*******
;  1290     2803  4 !******* PROTECTED
;  1291     2804  4 !*******
;  1292     2805  4
;  1293     2806  4          !+
```

```
; 1294          2807  4          !  Was terminator seen on PROTECTED READ?
; 1295          2808  4          !
; 1296          2809  4          !  Looking for terminator to make sure that user hasn't tried to go
; 1297          2810  4          !  beyond the bounds of a PROTECTED READ.
; 1298          2811  4          !
; 1299          2812  4          !  Two ways for a protected read to complete -
; 1300          2813  4          !       1. terminator typed before buffer filled ( no further check necessary)
; 1301          2814  4          !       2. buffer fill ( no terminator seen)
; 1302          2815  4          !               - do a one character read to make sure terminator is
; 1303          2816  4          !                       typed, not another character.
; 1304          2817  4          !
; 1305          2818  4          !  The following RAB fields look like this if buffer filled
; 1306          2819  4          !               rab [rab$l_sts]        = status
; 1307          2820  4          !               rab [rab$l_rsz]        = x       no. of chars read (acc_size)
; 1308          2821  4          !               rab [cob$$b_stv0_term] = 0       no terminator seen
; 1309          2822  4          !               rab [cob$$b_stv2_len]  = 0       size of terminator
; 1310          2823  4          !-
; 1311          2824  4
; 1312          2825  5          IF (.YES_PROTECT )                                  ! Was PROTECTION requested?
; 1313          2826  5                AND  ( .CHARS_READ NEQ 0 )                    ! AND is it needed
; 1314          2827  4          THEN
; 1315          2828  4          IF .RAB [RAB$L_STS] EQL RMS$_TNS                    ! RMS$_TNS = Terminator
; 1316          2829  4                              AND .TERM_SEEN EQL 0            ! Not Seen
; 1317          2830  4           THEN
; 1318          2831  5               BEGIN                                          ! Begin protect check $GET
; 1319          2832  5               !+
; 1320          2833  5               !  After initial $GET is performed it is necessary to perform a
; 1321          2834  5               !  $GET of length 1 to make sure that there are no characters
; 1322          2835  5               !  typed by the user that exceed the maximum allowed.
; 1323          2836  5               !  (Do not echo character to terminal.)
; 1324          2837  5               !  If the $GET of one character results in a terminator, there
; 1325          2838  5               !  is no problem.
; 1326          2839  5               !  If the $GET of one character results in an attempt to type
; 1327          2840  5               !  extra characters, there is an error.
; 1328          2841  5               !
; 1329          2842  5               !  If VAX RPG is the caller, always return control to the
; 1330          2843  5               !  calling program on an error.
; 1331          2844  5               !-
; 1332          2845  5
; 1333          2846  5               LOCAL
; 1334          2847  5                   NO_CHAR    :  INITIAL (0),              ! =1 no Protection error
; 1335          2848  5                   HAVE_TERM  :  INITIAL (0) ;             ! =1 terminator seen
; 1336          2849  5
; 1337          2850  5               WHILE .HAVE_TERM NEQ 1 DO
; 1338          2851  6                   BEGIN                                   ! Begin HAVE_TERM loop
; 1339          2852  6
; 1340          2853  6                   NO_CHAR = 0 ;
; 1341          2854  6                   FUNC_VAL_2 = TRM$M_TM_ESCAPE + TRM$M_TM_NOFILTR
; 1342          2855  6                                + TRM$M_TM_TRMNOECHO + TRM$M_TM_NOECHO ;
; 1343          2856  6
; 1344          2857  6                   RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
; 1345          2858  6                   COB$$RMS_GET ( .RAB, .FUNC_VAL_2, 1, NEXT_CHAR ) ;
; 1346          2859  6
; 1347          2860  6                   !+
; 1348          2861  6                   !  If user did not attempt to enter more data, set TERM_SIZE
; 1349          2862  6                   !  and TERM_IN_NEXT before possible call to COB$$PARTIAL_SEQ.
; 1350          2863  6                   !  If not enough room in $GET buffer to hold entire escape
```

```
;  1351          2864  6                       !  sequence then call COB$$PARTIAL_SEQ to read remainder
;  1352          2865  6                       !  of sequence.
;  1353          2866  6                       !-
;  1354          2867  6
;  1355          2868  6                       IF .RAB [RAB$W_RSZ] EQL 0               ! No more data entered.
;  1356          2869  6                       THEN
;  1357          2870  7                           BEGIN
;  1358          2871  7                           NO_CHAR = 1 ;                       ! Move terminator into
;  1359          2872  7                           NEXT_CHAR [0] = .RAB [COB$$B_STV0_TERM] ; ! NEXT_CHAR
;  1360          2873  6                           END ;
;  1361          2874  6                       TERM_SIZE = .RAB [COB$$B_STV2_LEN] ; ! Terminator size.
;  1362          2875  6                       TERM_LOC  = .RAB [COB$$B_STV0_TERM]; ! Terminator location.
;  1363          2876  6                       TERM_IN_NEXT = 1 ;                  ! Terminator on NEXT_CHAR
;  1364          2877  6                       IF .RAB [RAB$L_STS] EQL RMS$_PES
;  1365          2878  6                       THEN
;  1366          2879  6                           COB$$PARTIAL_SEQ ( PARAMETERS, .UNIT ) ;
;  1367          2880  6
;  1368          2881  6                       !+
;  1369          2882  6                       !  Terminators are the only acceptable input at this point.
;  1370          2883  6                       !  If NO_CHAR = 1 then there is no Protection error.
;  1371          2884  6                       !-
;  1372          2885  6
;  1373          2886  6                       IF .NO_CHAR
;  1374          2887  6                       THEN
;  1375          2888  7                           BEGIN                           ! Begin TERM accepted
;  1376          2889  7                           PROT_OK      = 1 ;              ! $GET successful
;  1377          2890  7                           HAVE_TERM    = 1 ;
;  1378          2891  7
;  1379          2892  7 !******
;  1380          2893  7 !****** DELETE KEY
;  1381          2894  7 !******
;  1382          2895  7                       !+
;  1383          2896  7                       !  Was termintor the DELETE KEY ?  If so, call
;  1384          2897  7                       !  COB$$DELETE_KEY to erase the last character
;  1385          2898  7                       !  read and to continue reading for input.
;  1386          2899  7                       !-
;  1387          2900  7
;  1388          2901  7                       IF .RAB [COB$$B_STV0_TERM] EQL DEL_KEY
;  1389          2902  7                       THEN
;  1390          2903  8                           BEGIN
;  1391          2904  8                           COB$$DELETE_KEY ( PARAMETERS, .UNIT, .FLAGS ) ;
;  1392          2905  8                           !+
;  1393          2906  8                           !  Check to see if we fell out of COB$$DELETE_KEY
;  1394          2907  8                           !  without a valid terminator.  If so, keep
;  1395          2908  8                           !  looking for it.
;  1396          2909  8                           !-
;  1397          2910  8                           IF .TERM_SIZE EQL 0
;  1398          2911  8                           THEN
;  1399          2912  9                               BEGIN
;  1400          2913  9                               HAVE_TERM = 0 ;             ! Loop again
;  1401          2914  9                               PROT_OK = 0 ;
;  1402          2915  9                               END
;  1403          2916  8                           ELSE
;  1404          2917  9                               BEGIN
;  1405          2918  9                               HAVE_TERM = 1 ;
;  1406          2919  9                               TERM_IN_NEXT = 0 ;          ! Note - COB$$DELETE_KEY put
;  1407          2920  8                               END ;                       ! the terminator in
```

```
;  1408          2921  8                                                   ! PUT_HERE.
;  1409          2922  7                           END ;
;  1410          2923  7
;  1411          2924  7                       END                         ! End TERM accepted
;  1412          2925  6                   ELSE
;  1413          2926  6
;  1414          2927  6 !*******
;  1415          2928  6 !******* PROTECTION ERROR
;  1416          2929  6 !*******
;  1417          2930  6
;  1418          2931  6                       !+
;  1419          2932  6                       ! PROTECTION error :
;  1420          2933  6                       ! User tried to input too many characters,
;  1421          2934  6                       !    - sound terminal bell,
;  1422          2935  6                       !    - leave cursor where it is (No reprompt or backspace).
;  1423          2936  6                       !-
;  1424          2937  6
;  1425          2938  7                           BEGIN
;  1426          2939  7                           COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
;  1427          2940  7                           PROT_OK = 0 ;               ! Signal Protection error
;  1428          2941  7                           HAVE_TERM = 0 ;
;  1429          2942  7                           END
;  1430          2943  7
;  1431          2944  5                       END ;                           ! End HAVE_TERM loop
;  1432          2945  5                   END                                 ! End protect check $GET
;  1433          2946  4               ELSE
;  1434          2947  4                   !+
;  1435          2948  4                   ! Protection requested but terminator already seen,
;  1436          2949  4                   ! no need for 1 character Read.
;  1437          2950  4                   !-
;  1438          2951  4                   PROT_OK = 1
;  1439          2952  4           ELSE
;  1440          2953  4               !+
;  1441          2954  4               ! Protection not requested, no need for 1 character Read.
;  1442          2955  4               !-
;  1443          2956  4               PROT_OK = 1 ;
;  1444          2957  4
;  1445          2958  4 !*******
;  1446          2959  4 !******* CONTROL KEY
;  1447          2960  4 !*******
;  1448          2961  4
;  1449          2962  4     IF .PROT_OK
;  1450          2963  4     THEN
;  1451          2964  4         !+
;  1452          2965  4         ! No sense going thru Control key code if there was a protection error.
;  1453          2966  4         !-
;  1454          2967  5         BEGIN                                           ! Begin Control Key
;  1455          2968  5
;  1456          2969  5             IF .TERM_IN_NEXT                            ! Locate terminator,
;  1457          2970  5             THEN                                        ! which buffer is it in.
;  1458          2971  5                 TERM_PTR = NEXT_CHAR[0]
;  1459          2972  5             ELSE
;  1460          2973  5                 TERM_PTR = .PUT_HERE[DSC$A_POINTER] + .CHARS_READ ;
;  1461          2974  5
;  1462          2975  5             !+
;  1463          2976  5             ! If parameter KEY not sent (.KEY = 0) then CR, TAB, CONTROL Z,
;  1464          2977  5             ! and DELETE KEY are the only legal terminators.
```

```
;  1465          2978  5               !
;  1466          2979  5               !  If parameter KEY not 0 then CR, TAB, CONTROL Z, DELETE KEY, PF,
;  1467          2980  5               !  ARROW and SPECIAL FUNCTION PROFESSIONAL Keys are legal
;  1468          2981  5               !  terminators.  Copy terminator to KEY parameter.
;  1469          2982  5               !
;  1470          2983  5               !  Special treatment needed for CONTROL Z under RMS.  There is a
;  1471          2984  5               !  difference between ^Z being typed alone and with data.
;  1472          2985  5               !  When ^Z is typed with data the ^Z is stored in
;  1473          2986  5               !  RAB[RAB$_STV0_TERM], but when ^Z is typed alone the status
;  1474          2987  5               !  RMS$_EOF is returned from the $Get Service.
;  1475          2988  5               !-
;  1476          2989  5
;  1477          2990  5               IF .TERM_SIZE EQL 1                             ! One byte terminator
;  1478          2991  5               THEN
;  1479          2992  6                   BEGIN
;  1480          2993  6                   TERM_PTR = RAB [COB$$B_STV0_TERM] ;
;  1481          2994  6                   SELECTONE .RAB [COB$$B_STV0_TERM] OF
;  1482          2995  6                       SET
;  1483          2996  6                           [ CR,                               ! Carriage Return
;  1484          2997  6                             TAB ] :                           ! Tab
;  1485          2998  6                               !+
;  1486          2999  6                               !  These keys are legal, do nothing if KEY = 0.
;  1487          3000  6                               !-
;  1488          3001  6                               IF NOT NULLPARAMETER (KEY)
;  1489          3002  6                               THEN
;  1490          3003  6                                   CH$MOVE ( 1, .TERM_PTR, .KEY [DSC$A_POINTER] ) ;
;  1491          3004  6
;  1492          3005  6                           [ CZ ] :                          ! Control z
;  1493          3006  6
;  1494          3007  6                               !+
;  1495          3008  6                               !  CONTROL Z hit along with data  ( terminator in
;  1496          3009  6                               !  RAB [COB$$B_STV0_TERM] )
;  1497          3010  6                               !-
;  1498          3011  6
;  1499          3012  7                               BEGIN
;  1500          3013  7                               IF (.FLAGS AND V_COB_RPG) NEQ 0
;  1501          3014  7                               THEN
;  1502          3015  8                                   BEGIN
;  1503          3016  8                                   !+
;  1504          3017  8                                   !  VAX RPG - Control Z is an illegal terminator.
;  1505          3018  8                                   !-
;  1506          3019  8                                   LEGAL = 0 ;
;  1507          3020  8                                   COB$$ILLEGAL_TERM ( PARAMETERS, .UNIT, .FLAGS,
;  1508          3021  8                                                                   .KEY ) ;
;  1509          3022  8                                   END
;  1510          3023  7                               ELSE
;  1511          3024  8                                   BEGIN
;  1512          3025  8                                   !+
;  1513          3026  8                                   !  VAX COBOL - Control Z has special meaning.
;  1514          3027  8                                   !-
;  1515          3028  8                                   COB$$CLEAN_UP ( PARAMETERS, .FLAGS ) ;
;  1516          3029  8                                   COB$$CONTROL_Z ( .UNIT, .KEY ) ;
;  1517          3030  8                                   RETURN 0 ;
;  1518          3031  7                                   END ;
;  1519          3032  6                               END ;
;  1520          3033  6
;  1521          3034  6                       [ DEL_KEY ] :                         ! Delete key
```

```
;  1522          3035  6
;  1523          3036  7                                 BEGIN
;  1524          3037  7                                 COB$$DELETE_KEY ( PARAMETERS, .UNIT, .FLAGS ) ;
;  1525          3038  6                                 END ;
;  1526          3039  6
;  1527          3040  6                             [OTHERWISE] :                       ! Error - key not a
;  1528          3041  6                                                                 ! terminator
;  1529          3042  7                                     BEGIN
;  1530          3043  7                                     LEGAL = 0 ;
;  1531          3044  7                                     COB$$ILLEGAL_TERM ( PARAMETERS, .UNIT, .FLAGS,
;  1532          3045  7                                                                                 .KEY ) ;
;  1533          3046  6                                     END ;
;  1534          3047  6                         TES ;
;  1535          3048  6                     END
;  1536          3049  6
;  1537          3050  5                 ELSE
;  1538          3051  5                     IF .CHARS_READ EQL 0  AND  .RAB [RAB$L_STS] EQL RMS$_EOF
;  1539          3052  5                     THEN
;  1540          3053  6                         BEGIN
;  1541          3054  6                         !+
;  1542          3055  6                         !  CONTROL Z hit alone - terminator not placed in
;  1543          3056  6                         !  RAB [COB$$B_STV0_TERM], but signaled via RAB [RAB$L_STS].
;  1544          3057  6                         !-
;  1545          3058  6                         IF (.FLAGS AND V_COB_RPG) NEQ 0
;  1546          3059  6                         THEN
;  1547          3060  7                             BEGIN
;  1548          3061  7                             LEGAL = 0 ;                     ! VAX RPG - Control Z
;  1549          3062  7                             COB$$ILLEGAL_TERM ( PARAMETERS, .UNIT, .FLAGS, .KEY ) ;
;  1550          3063  7                             END
;  1551          3064  6                         ELSE
;  1552          3065  7                             BEGIN                           ! VAX COBOL
;  1553          3066  7                             COB$$CLEAN_UP ( PARAMETERS, .FLAGS ) ;
;  1554          3067  7                             COB$$CONTROL_Z ( .UNIT, .KEY ) ;
;  1555          3068  7                             RETURN 0 ;
;  1556          3069  6                             END ;
;  1557          3070  6                         END
;  1558          3071  5                 ELSE
;  1559          3072  6                     BEGIN
;  1560          3073  6
;  1561          3074  6                     !+
;  1562          3075  6                     !  Escape Sequence as Terminator.  .TERM_SIZE greater
;  1563          3076  6                     !  than 1 and RMS_$EOF not signaled.
;  1564          3077  6                     !-
;  1565          3078  6
;  1566          3079  6                     IF NOT NULLPARAMETER (KEY)
;  1567          3080  6                     THEN
;  1568          3081  6                         !+
;  1569          3082  6                         !  COB$$CONTROL_KEY converts terminator sequences to
;  1570          3083  6                         !  COBOL defined sequences and fills in KEY parameter
;  1571          3084  6                         !  if terminator is legal.
;  1572          3085  6                         !-
;  1573          3086  7                         BEGIN
;  1574          3087  8                         IF NOT ( COB$$CONTROL_KEY (TERM_PTR, .TERM_SIZE, .KEY) )
;  1575          3088  7                         THEN
;  1576          3089  8                             BEGIN
;  1577          3090  8                             LEGAL = 0 ;                 ! Illegal escape sequence
;  1578          3091  8                             COB$$ILLEGAL_TERM ( PARAMETERS, .UNIT, .FLAGS,
```

```
; 1579          3092  8                                                                                   .KEY ) ;
; 1580          3093  7                                 END ;
; 1581          3094  7                             END
; 1582          3095  6                         ELSE
; 1583          3096  6                             !+
; 1584          3097  6                             ! Terminator of size greater than 1 is illegal when KEY is
; 1585          3098  6                             ! not used.
; 1586          3099  6                             !-
; 1587          3100  7                             BEGIN
; 1588          3101  7                             LEGAL = 0 ;
; 1589          3102  7                             COB$$ILLEGAL_TERM ( PARAMETERS, .UNIT, .FLAGS, .KEY ) ;
; 1590          3103  6                             END ;
; 1591          3104  5                         END ;
; 1592          3105  4             END ;                                               ! End Control Key
; 1593          3106  4
; 1594          3107  4             !*******
; 1595          3108  4             !******* NULL INPUT
; 1596          3109  4             !*******
; 1597          3110  4
; 1598          3111  4                     !+
; 1599          3112  4                     ! Null input
; 1600          3113  4                     ! RAB fields look like this for null input
; 1601          3114  4                     !           rab [rab$l_sts]        = 1      status
; 1602          3115  4                     !           rab [rab$l_rsz]        = 0      no. of chars read
; 1603          3116  4                     !           rab [cob$$b_stv0_term] = d      <cr> terminator seen
; 1604          3117  4                     !           rab [cob$$b_stv2_len]  = 1      size of terminator
; 1605          3118  4                     !
; 1606          3119  4                     ! Check for DEFAULT parameter - if present prepare to put it through
; 1607          3120  4                     ! Conversion routines by placing DEFAULT in PUT_HERE.
; 1608          3121  4                     !-
; 1609          3122  4
; 1610          3123  5                     IF ( .CHARS_READ EQL 0 ) AND (( .FLAGS AND V_COB_RPG ) NEQ 0 )
; 1611          3124  4                     THEN
; 1612          3125  4                         !+
; 1613          3126  4                         ! In case of null input for RPG, simply return (no DEFAULT).
; 1614          3127  4                         ! But perform any necessary clean up first.
; 1615          3128  4                         !-
; 1616          3129  5                         BEGIN
; 1617          3130  5                         COB$$RPG_CLEAN_UP ( .FLAGS ) ;
; 1618          3131  5                         RETURN 1 ;
; 1619          3132  4                         END ;
; 1620          3133  4
; 1621          3134  5                     IF ( .CHARS_READ EQL 0 )
; 1622          3135  4                     THEN
; 1623          3136  5                         IF NOT NULLPARAMETER (DEFAULT)  AND  (.YES_DEFAULT EQL 0)
; 1624          3137  4                         THEN
; 1625          3138  5                             BEGIN                                   ! Begin DEFAULT
; 1626          3139  5
; 1627          3140  5                             CHARS_READ = .DEFAULT [DSC$W_LENGTH];
; 1628          3141  5                             YES_DEFAULT = 1 ;
; 1629          3142  5
; 1630          3143  5                             !+
; 1631          3144  5                             ! Protection check for DEFAULT excluding the Floating
; 1632          3145  5                             ! Point data types ( these will be handled in routine
; 1633          3146  5                             ! COB$$VERIFY_FL_RANGE ).
; 1634          3147  5                             !-
; 1635          3148  5
```

```
;  1636     3149  6                 IF (.YES_PROTECT AND
;  1637     3150  7                         ( .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_F AND
;  1638     3151  6                           .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_D ))
;  1639     3152  5                 THEN
;  1640     3153  5                     !+
;  1641     3154  5                     !  If the length of DEFAULT is greater than the expected
;  1642     3155  5                     !  input size ACC_SIZE, then there is a Protection error.
;  1643     3156  5                     !  This should be caught at compile time by VAX COBOL and
;  1644     3157  5                     !  a fatal error message issued, however there is one case
;  1645     3158  5                     !  that the compiler cannot catch, therefore issue a fatal
;  1646     3159  5                     !  run time error here.
;  1647     3160  5                     !-
;  1648     3161  6                     IF (.DEFAULT [DSC$W_LENGTH] GTR .ACC_SIZE)
;  1649     3162  5                     THEN
;  1650     3163  5                         LIB$STOP ( COB$_INVDEFVAL )
;  1651     3164  5                     ELSE
;  1652     3165  5                         PROT_OK = 1                         ! No PROTECT error
;  1653     3166  5                 ELSE
;  1654     3167  5                     PROT_OK = 1 ;                           ! No PROTECT error
;  1655     3168  5
;  1656     3169  4                 END ;                                       ! End DEFAULT
;  1657     3170  4
;  1658     3171  4             !*******
;  1659     3172  4             !******* CONVERSION
;  1660     3173  4             !******* ALL RMS $GETs COMPLETED EXCEPT POSSIBLE REPROMPT ON A CONVERSION ERROR.
;  1661     3174  4             !*******
;  1662     3175  4
;  1663     3176  4             !+
;  1664     3177  4             !  If conversion requested, call routine COB$$ACC_CONVERT
;  1665     3178  4             !-
;  1666     3179  4
;  1667     3180  5                 IF ( .PROT_OK )                                 ! If protection error,
;  1668     3181  4                 THEN                                            ! don't go thru conversion
;  1669     3182  5                     IF ( .YES_CONV )
;  1670     3183  4                     THEN
;  1671     3184  4                         CONV_OK = COB$$ACC_CONVERT ( .STRING_DEST, .FLAGS,
;  1672     3185  4                                                      .DEFAULT, PUT_HERE, .CHARS_READ,
;  1673     3186  4                                                      .YES_DEFAULT, .YES_SIGN )
;  1674     3187  4                     ELSE
;  1675     3188  5                         BEGIN
;  1676     3189  5                             LOCAL
;  1677     3190  5                                 COPY_NUM ;
;  1678     3191  5                             !+
;  1679     3192  5                             !  No conversion requested - copy input data to STRING_DEST.
;  1680     3193  5                             !  Use STR$COPY_R because it BLANK fills.
;  1681     3194  5                             !-
;  1682     3195  5
;  1683     3196  5                             IF .CHARS_READ LSS .STRING_DEST[DSC$W_LENGTH]
;  1684     3197  5                             THEN
;  1685     3198  5                                 COPY_NUM = .CHARS_READ
;  1686     3199  5                             ELSE
;  1687     3200  5                                 COPY_NUM = .STRING_DEST[DSC$W_LENGTH] ;
;  1688     3201  5
;  1689     3202  5                             STR$COPY_R ( .STRING_DEST, COPY_NUM,
;  1690     3203  6                                          (IF .YES_DEFAULT
;  1691     3204  6                                           THEN .DEFAULT [DSC$A_POINTER]
;  1692     3205  5                                           ELSE .PUT_HERE [DSC$A_POINTER] )) ;
```

```
; 1693          3206  5
; 1694          3207  5
; 1695          3208  5                         CONV_OK = 1 ;                           ! set CONV_OK to success
; 1696          3209  4                     END;
; 1697          3210  4
; 1698          3211  4                 !+
; 1699          3212  4                 ! Conversion completed - was it successful ?
; 1700          3213  4                 !-
; 1701          3214  4
; 1702          3215  4                 IF .CONV_OK EQL 0
; 1703          3216  4                 THEN
; 1704          3217  4                 !+
; 1705          3218  4                 ! CONVERSION error.  Read UNIT parameter to determine what
; 1706          3219  4                 ! to do.
; 1707          3220  4                 !
; 1708          3221  4                 !    Byte 2 of                      Conversion
; 1709          3222  4                 !      UNIT                           error
; 1710          3223  4                 !
; 1711          3224  4                 !       0                             reprompt
; 1712          3225  4                 !       1 ( at end )                  reprompt
; 1713          3226  4                 !       2 ( on exception )            return
; 1714          3227  4                 !-
; 1715          3228  5                 BEGIN                                   ! Begin conversion error
; 1716          3229  5                     IF ( .FLAGS AND V_COB_RPG ) NEQ 0
; 1717          3230  5                     THEN
; 1718          3231  5                         !+
; 1719          3232  5                         ! VAX RPG - return on a Conversion Error, ring bell
; 1720          3233  5                         ! and clean up before exiting.
; 1721          3234  5                         !-
; 1722          3235  6                         BEGIN
; 1723          3236  6                         COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
; 1724          3237  6                         COB$$RPG_CLEAN_UP ( .FLAGS ) ;
; 1725          3238  6                         RETURN 0 ;
; 1726          3239  5                         END ;
; 1727          3240  5
; 1728          3241  5                     IF .UNIT [1] EQL 2
; 1729          3242  5                     THEN
; 1730          3243  6                         BEGIN
; 1731          3244  6                         !+
; 1732          3245  6                         ! Clean up before returning control to VAX COBOL.
; 1733          3246  6                         !-
; 1734          3247  6                         COB$$CLEAN_UP ( PARAMETERS, .FLAGS ) ;
; 1735          3248  6                         RETURN 0 ;
; 1736          3249  6                         END
; 1737          3250  5                     ELSE
; 1738          3251  5                         !+
; 1739          3252  5                         ! Reprompt
; 1740          3253  5                         !    - sound terminal bell,
; 1741          3254  5                         !    - clear screen of all typed characters,
; 1742          3255  5                         !    - reset cursor to original line/column position
; 1743          3256  5                         !-
; 1744          3257  6                         BEGIN
; 1745          3258  7                         IF ((.FLAGS AND V_NO_ECHO) NEQ 0 )  OR  (.YES_DEFAULT )
; 1746          3259  6                         THEN
; 1747          3260  7                             BEGIN                           ! No re-positioning
; 1748          3261  7                             COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;  ! necessary
; 1749          3262  7                             END
```

```
; 1750          3263  6                       ELSE
; 1751          3264  7                           BEGIN
; 1752          3265  7                           $ERROR_REPROMPT ;               ! Re-position
; 1753          3266  6                           END ;
; 1754          3267  6                       CONV_OK = 0 ;                       ! Signal Conversion error
; 1755          3268  6                       YES_DEFAULT = 0 ;
; 1756          3269                          END ;
; 1757          3270                  END                                         ! End conversion error
; 1758          3271
; 1759          3272              ELSE
; 1760          3273                  !+
; 1761          3274                  ! Conversion not done either because Protection failed (PROT_OK=0)
; 1762          3275                  ! or there was no data to convert
; 1763          3276                  !-
; 1764          3277                  CONV_OK = 1 ;
; 1765          3278
; 1766          3279          END ;                                               ! End loop
; 1767          3280
; 1768          3281              !+
; 1769          3282              ! RMS $GET complete - fill in optional LENGTH parameter with the
; 1770          3283              ! number of characters read.
; 1771          3284              !-
; 1772          3285
; 1773          3286              IF NOT NULLPARAMETER (LENGTH)
; 1774          3287              THEN
; 1775          3288                  .LENGTH = .CHARS_READ ;
; 1776          3289
; 1777          3290      !*******
; 1778          3291      !******* CLEAN UP
; 1779          3292      !*******
; 1780          3293
; 1781          3294      !+
; 1782          3295      ! Call COB$$CLEAN_UP to perform (if needed) cursor positioning,
; 1783          3296      ! turn off terminal attributes, and advancing.
; 1784          3297      !-
; 1785          3298
; 1786          3299          COB$$CLEAN_UP ( PARAMETERS, .FLAGS ) ;
; 1787          3300
; 1788          3301          END;                                                !  End $GET
; 1789          3302
; 1790          3303      !+
; 1791          3304      ! Free local strings PUT_HERE
; 1792          3305      !-
; 1793          3306
; 1794          3307          IF NOT ( STR$FREE1_DX ( PUT_HERE ))
; 1795          3308          THEN LIB$STOP ( COB$_ERRDURACC ) ;
; 1796          3309
; 1797          3310
; 1798          3311          RETURN 1;
; 1799          3312  1       END;                                          ! end of routine COB$ACC_SCR


                                                001DE                .BLKB   2
                                    00000000# 001E0 P.AAR:           .LONG   0[22]                                              ;
```

```
                                        OFFC 00000          .ENTRY  COB$ACC_SCR, Save R2,R3,R4,R5,R6,R7,R8,R9,-   ; 2181
                                                                    R10,R11                                       ;
                        5E    EBF0   CE   9E 00002          MOVAB   -5136(SP), SP                                 ;
                                0C   AE   D4 00007          CLRL    ON_LEN                                        ; 2271
                                     5A   D4 0000A          CLRL    PROT_OK                                       ;
                                     7E   7C 0000C          CLRQ    REPROMPT_DONE                                 ;
                                     7E   7C 0000E          CLRQ    YES_NO_ECHO                                   ;
                                     01   DD 00010          PUSHL   #1                                            ;
                                     56   D4 00012          CLRL    P_DATA_TYPE                                   ;
        94   AD    8E   AF    0058   8F   28 00014          MOVC3   #88, P.AAR, PARAMETERS                        ; 2297
                                     50   D4 0001C          CLRL    ZEROES                                        ; 2312
                   24   AE           20   D0 0001E          MOVL    #32, BLANKS                                   ; 2313
                   94   AD 02100000  8F   D0 00022          MOVL    #34603008, PUT_HERE                           ; 2319
                                98   AD   D4 0002A          CLRL    PUT_HERE+4                                    ; 2322
                   59           0C   AC   D0 0002D          MOVL    FLAGS, R9                                     ; 2337
             04         59           05   E1 00031          BBC     #5, R9, 1$                                    ;
                   08   AE           01   D0 00035          MOVL    #1, YES_CONV                                  ;
                                     59   95 00039 1$:      TSTB    R9                                            ; 2338
                                     02   18 0003B          BGEQ    2$                                            ;
                                     6E   D4 0003D          CLRL    YES_SIGN                                      ;
             03         59           08   E0 0003F 2$:      BBS     #8, R9, 3$                                    ; 2340
                                   00FC   31 00043          BRW     20$                                           ;
                   D0   AD           01   D0 00046 3$:      MOVL    #1, YES_PROTECT                               ; 2343
                                14   AC   D5 0004A          TSTL    SIZE                                          ; 2344
                                     08   13 0004D          BEQL    5$                                            ;
                   AC   AD      14   AC   B0 0004F          MOVW    SIZE, ACC_SIZE                                ; 2345
                                   00F1   31 00054 4$:      BRW     21$                                           ;
                                     52   D4 00057 5$:      CLRL    PP99                                          ; 2347
                        50      08   AC   D0 00059          MOVL    STRING_DEST, R0                               ; 2352
                        52      09   A0   9A 0005D          MOVZBL  9(R0), PP99                                   ;
                        51      08   A0   98 00061          CVTBL   8(R0), R1                                     ;
                        52           51   C0 00065          ADDL2   R1, PP99                                      ;
                   AC   AD           60   B0 00068          MOVW    (R0), ACC_SIZE                                ; 2353
                                     54   D4 0006C          CLRL    R4                                            ; 2361
                        09      03   A0   91 0006E          CMPB    3(R0), #9                                     ;
                                     6B   12 00072          BNEQ    13$                                           ;
                                     54   D6 00074          INCL    R4                                            ;
                                     52   D5 00076          TSTL    PP99                                          ; 2362
                                     05   19 00078          BLSS    6$                                            ;
                                08   A0   95 0007A          TSTB    8(R0)                                         ; 2363
                                     60   15 0007D          BLEQ    13$                                           ;
                        56           01   D0 0007F 6$:      MOVL    #1, P_DATA_TYPE                               ; 2366
                                     53   D4 00082          CLRL    R3                                            ; 2367
                                     52   D5 00084          TSTL    PP99                                          ;
                                     11   18 00086          BGEQ    8$                                            ;
                                     53   D6 00088          INCL    R3                                            ;
                        51      08   A0   98 0008A          CVTBL   8(R0), R1                                     ; 2369
                                     03   18 0008E          BGEQ    7$                                            ;
                        51           51   CE 00090          MNEGL   R1, R1                                        ;
                   AC   AD           51   B0 00093 7$:      MOVW    R1, ACC_SIZE                                  ;
                                     04   11 00097          BRB     9$                                            ;
                   AC   AD           52   B0 00099 8$:      MOVW    PP99, ACC_SIZE                                ; 2371
                        B3      08   AE   E9 0009D 9$:      BLBC    YES_CONV, 4$                                  ; 2373
                        03           53   E9 000A1          BLBC    R3, 10$                                       ; 2379
                                AC   AD   B6 000A4          INCW    ACC_SIZE                                      ; 2381
                        51      02   A0   9A 000A7 10$:     MOVZBL  2(R0), R1                                     ; 2389
```

```
             0F          51  91 000AB         CMPB    R1, #15                      ;
                         03  13 000AE         BEQL    11$                          ;
                   AC    AD  B6 000B0         INCW    ACC_SIZE                     ; 2391
             07          51  91 000B3  11$:   CMPB    R1, #7                       ; 2397
                         1E  13 000B6         BEQL    12$                          ;
             03          51  91 000B8         CMPB    R1, #3                       ; 2398
                         19  13 000BB         BEQL    12$                          ;
             08          51  91 000BD         CMPB    R1, #8                       ; 2399
                         14  13 000C0         BEQL    12$                          ;
             04          51  91 000C2         CMPB    R1, #4                       ; 2400
                         0F  13 000C5         BEQL    12$                          ;
             09          51  91 000C7         CMPB    R1, #9                       ; 2401
                         0A  13 000CA         BEQL    12$                          ;
             05          51  91 000CC         CMPB    R1, #5                       ; 2402
                         05  13 000CF         BEQL    12$                          ;
             15          51  91 000D1         CMPB    R1, #21                      ; 2403
                         72  12 000D4         BNEQ    21$                          ;
                         6E  D5 000D6  12$:   TSTL    YES_SIGN                     ; 2404
                         6E  12 000D8         BNEQ    21$                          ;
                   AC    AD  B7 000DA         DECW    ACC_SIZE                     ; 2406
                         69  11 000DD         BRB     21$                          ; 2361
             65    08    AE  E9 000DF  13$:   BLBC    YES_CONV, 21$                ; 2415
             51    02    A0  9A 000E3         MOVZBL  2(R0), R1                    ; 2423
             13          51  91 000E7         CMPB    R1, #19                      ;
                         0D  13 000EA         BEQL    14$                          ;
             11          51  91 000EC         CMPB    R1, #17                      ; 2424
                         08  13 000EF         BEQL    14$                          ;
             15          51  91 000F1         CMPB    R1, #21                      ; 2425
                         06  12 000F4         BNEQ    15$                          ;
             03          6E  E9 000F6         BLBC    YES_SIGN, 15$                ; 2426
                   AC    AD  B6 000F9  14$:   INCW    ACC_SIZE                     ; 2428
             29          54  E9 000FC  15$:   BLBC    R4, 17$                      ; 2438
             07          51  91 000FF         CMPB    R1, #7                       ; 2440
                         19  13 00102         BEQL    16$                          ;
             03          51  91 00104         CMPB    R1, #3                       ; 2441
                         14  13 00107         BEQL    16$                          ;
             08          51  91 00109         CMPB    R1, #8                       ; 2442
                         0F  13 0010C         BEQL    16$                          ;
             04          51  91 0010E         CMPB    R1, #4                       ; 2443
                         0A  13 00111         BEQL    16$                          ;
             09          51  91 00113         CMPB    R1, #9                       ; 2444
                         05  13 00116         BEQL    16$                          ;
             05          51  91 00118         CMPB    R1, #5                       ; 2445
                         0B  12 0011B         BNEQ    17$                          ;
             08          6E  E9 0011D  16$:   BLBC    YES_SIGN, 17$                ; 2446
       AC    AD    09    A0  9B 00120         MOVZBW  9(R0), ACC_SIZE              ; 2448
                   AC    AD  B6 00125         INCW    ACC_SIZE                     ;
             0A          51  91 00128  17$:   CMPB    R1, #10                      ; 2454
                         04  12 0012B         BNEQ    18$                          ;
       AC    AD          0D  B0 0012D         MOVW    #13, ACC_SIZE                ; 2456
             0B          51  91 00131  18$:   CMPB    R1, #11                      ; 2457
                         04  12 00134         BNEQ    19$                          ;
       AC    AD          16  B0 00136         MOVW    #22, ACC_SIZE                ; 2459
             0B          54  E9 0013A  19$:   BLBC    R4, 21$                      ; 2465
                   AC    AD  B6 0013D         INCW    ACC_SIZE                     ; 2467
                         06  11 00140         BRB     21$                          ; 2340
       AC    AD  03F2    8F  B0 00142  20$:   MOVW    #1010, ACC_SIZE              ; 2473
```

```
                        50        AC   AD   B0 00148 21$:    MOVW    ACC_SIZE, PUT_SIZE                      ; 2481
                    0398  8F        AC   AD   B1 0014C        CMPW    ACC_SIZE, #920                          ; 2482
                                           05   1E 00152        BGEQU   22$
                                           05   A1 00154        ADDW3   #5, ACC_SIZE, PUT_SIZE
                50        AC   AD                             
                                       94  AD   9F 00159 22$:    PUSHAB  PUT_HERE                                ; 2484
                      20  AE               50   3C 0015C        MOVZWL  PUT_SIZE, 32(SP)
                                       20  AE   9F 00160        PUSHAB  32(SP)
                00000000G 00               02   FB 00163        CALLS   #2, STR$GET1_DX
                          0D               50   E8 0016A        BLBS    R0, 23$
                                00000000G  8F   DD 0016D        PUSHL   #COB$_ERRDURACC                         ; 2485
                00000000G 00               01   FB 00173        CALLS   #1, LIB$STOP
                          52        04     AC   9A 0017A 23$:    MOVZBL  UNIT, R2                                ; 2493
                          06               52   91 0017E        CMPB    R2, #6
                                           0D   1B 00181        BLEQU   24$
                                00000000G  8F   DD 00183        PUSHL   #COB$_INVARG                            ; 2495
                00000000G 00               01   FB 00189        CALLS   #1, LIB$STOP
                      14  AE 00000000G0042   DE 00190 24$:    MOVAL   COB$$AL_WRITE_RAB[R2], 20(SP)           ; 2497
                                     14    BE   D5 00199        TSTL    @20(SP)
                                           11   12 0019C        BNEQ    27$
            04            59               0B   E1 0019E        BBC     #11, R9, 25$                            ; 2504
                                           01   DD 001A2        PUSHL   #1
                                           02   11 001A4        BRB     26$
                                           7E   D4 001A6 25$:    CLRL    -(SP)
                                           52   DD 001A8 26$:    PUSHL   R2                                      ; 2503
                    0000V CF               02   FB 001AA        CALLS   #2, COB$$OPEN_IN
                          57        14     BE   D0 001AF 27$:    MOVL    @20(SP), RAB                            ; 2508
                          50        44     A7   9E 001B3        MOVAB   68(R7), NAM_DSC                         ; 2519
                                00000000G  00   D5 001B7        TSTL    COB$ACC_TERM_TYPE                       ; 2521
                                           23   12 001BD        BNEQ    28$
                                00000000G  00   9F 001BF        PUSHAB  COB$ACC_TERM_TYPE                       ; 2523
                             7E            60   3C 001C5        MOVZWL  (NAM_DSC), -(SP)                        ; 2524
                                       04  A0   DD 001C8        PUSHL   4(NAM_DSC)                              ; 2523
                00000000G 00               03   FB 001CB        CALLS   #3, COB$$SETUP_TERM_TYPE
                          0D               50   E8 001D2        BLBS    R0, 28$
                                00000000G  8F   DD 001D5        PUSHL   #COB$_ERRDURACC                         ; 2526
                00000000G 00               01   FB 001DB        CALLS   #1, LIB$STOP
                                00000000G  00   D5 001E2 28$:    TSTL    COB$ACC_TERM_TYPE                       ; 2528
                                           46   12 001E8        BNEQ    31$
                                           6E   DD 001EA        PUSHL   YES_SIGN                                ; 2539
                                     D0    AD   DD 001EC        PUSHL   YES_PROTECT
                                     10    AE   DD 001EF        PUSHL   YES_CONV                                ; 2538
                                     94    AD   9F 001F2        PUSHAB  PUT_HERE                                ; 2537
                             7E            AC   AD 3C 001F5    MOVZWL  ACC_SIZE, -(SP)                         ; 2538
                                     1C    AC   DD 001F9        PUSHL   LENGTH
                                     10    AC   DD 001FC        PUSHL   DEFAULT                                 ; 2537
                                           59   DD 001FF        PUSHL   R9
                             7E            04   AC 7D 00201    MOVQ    UNIT, -(SP)
                          0000V CF               0A   FB 00205    CALLS   #10, COB$$ACC_SCR_FILE
                                52               50   D0 0020A    MOVL    R0, STATUS
                                       94  AD   9F 0020D        PUSHAB  PUT_HERE                                ; 2543
                00000000G 00               01   FB 00210        CALLS   #1, STR$FREE1_DX
                          0D               50   E8 00217        BLBS    R0, 29$
                                00000000G  8F   DD 0021A        PUSHL   #COB$_ERRDURACC                         ; 2544
                00000000G 00               01   FB 00220        CALLS   #1, LIB$STOP
                          03               52   E8 00227 29$:    BLBS    STATUS, 30$                             ; 2546
                                      0542 31   0022A        BRW     106$
                                      053B 31   0022D 30$:    BRW     105$
```

```
                    00000000'  EF            01  D0 00230 31$:   MOVL    #1, ACC_SCR                       ; 2560
                           50  00000000G     00  9A 00237        MOVZBL  COB$$AB_PREV, R0                  ; 2575
                                             0A  13 0023E        BEQL    32$                               ;
                           02                50  91 00240        CMPB    R0, #2                            ; 2576
                                             05  13 00243        BEQL    32$                               ;
                           04                50  91 00245        CMPB    R0, #4                            ; 2577
                                             09  12 00248        BNEQ    33$                               ;
                                             59  DD 0024A 32$:   PUSHL   R9                                ; 2582
                                             01  DD 0024C        PUSHL   #1                                ;
                        0000V  CF            02  FB 0024E        CALLS   #2, COB$$RMS_PUT_BYTE             ;
    D8   AD          59        04            00  EF 00253 33$:   EXTZV   #0, #4, R9, PUT_FLAG              ; 2591
                                             2C  13 00259        BEQL    34$                               ; 2593
                                    E8       AD  9F 0025B        PUSHAB  OFF_LEN                           ; 2597
                                    DC       AD  9F 0025E        PUSHAB  OFF_BUF                           ;
                                    28       AE  9F 00261        PUSHAB  ON_LEN                            ; 2596
                                    EC       AD  9F 00264        PUSHAB  ON_BUF                            ;
                                    D8       AD  DD 00267        PUSHL   PUT_FLAG                          ; 2595
                              00000000G      00  DD 0026A        PUSHL   COB$ACC_TERM_TYPE                 ;
                    00000000G  00            06  FB 00270        CALLS   #6, COB$$SET_ATTRIBUTES_ONLY      ;
                               0D            50  E8 00277        BLBS    R0, 34$                           ;
                              00000000G      8F  DD 0027A        PUSHL   #COB$_ERRDURACC                   ; 2598
                    00000000G  00            01  FB 00280        CALLS   #1, LIB$STOP                      ;
                     0C        59            04  E1 00287 34$:   BBC     #4, R9, 35$                       ; 2604
                               50   EC       AD  9E 0028B        MOVAB   ON_BUF, R0                        ; 2607
                         20 BE40             07  90 0028F        MOVB    #7, @ON_LEN[R0]                   ;
                                    20       AE  D6 00294        INCL    ON_LEN                            ; 2608
                               06            6C  91 00297 35$:   CMPB    (AP), #6                          ; 2618
                                             58  1F 0029A        BLSSU   41$                               ;
                                    18       AC  D5 0029C        TSTL    24(AP)                            ;
                                             53  13 0029F        BEQL    41$                               ;
                         28    AE   18       BC  3C 002A1        MOVZWL  @KEY, KEY_LEN                     ; 2624
                                    24       AE  9F 002A6        PUSHAB  BLANKS                            ; 2625
                                    2C       AE  9F 002A9        PUSHAB  KEY_LEN                           ;
                                    18       AC  DD 002AC        PUSHL   KEY                               ;
                    00000000G  00            03  FB 002AF        CALLS   #3, STR$DUPL_CHAR                 ;
                               07            6C  91 002B6        CMPB    (AP), #7                          ; 2632
                                             05  1F 002B9        BLSSU   36$                               ;
                                    1C       AC  D5 002BB        TSTL    28(AP)                            ;
                                             34  12 002BE        BNEQ    41$                               ;
                               05            6C  91 002C0 36$:   CMPB    (AP), #5                          ; 2633
                                             05  1F 002C3        BLSSU   37$                               ;
                                    14       AC  D5 002C5        TSTL    20(AP)                            ;
                                             2A  12 002C8        BNEQ    41$                               ;
                               04            6C  91 002CA 37$:   CMPB    (AP), #4                          ; 2634
                                             05  1F 002CD        BLSSU   38$                               ;
                                    10       AC  D5 002CF        TSTL    16(AP)                            ;
                                             20  12 002D2        BNEQ    41$                               ;
                               02            6C  91 002D4 38$:   CMPB    (AP), #2                          ; 2635
                                             05  1F 002D7        BLSSU   39$                               ;
                                    08       AC  D5 002D9        TSTL    8(AP)                             ;
                                             16  12 002DC        BNEQ    41$                               ;
                                    18       AC  DD 002DE 39$:   PUSHL   KEY                               ; 2637
                                             59  DD 002E1        PUSHL   R9                                ;
                                    04       AC  DD 002E3        PUSHL   UNIT                              ;
                        0000V  CF            03  FB 002E6        CALLS   #3, COB$$FORMAT_FOUR              ;
                               03            50  E8 002EB        BLBS    R0, 40$                           ;
                                           047E  31 002EE        BRW     106$                              ;
```

```
                                                  0477  31 002F1 40$:     BRW     105$                                    ;
                                                    18  AE  D4 002F4 41$:     CLRL    24(SP)                                  ; 2653
                                                    0F  59  09 E1 002F7         BBC     #9, R9, 42$                             ;
                                                    18  AE  D6 002FB         INCL    24(SP)                                  ;
                                                B4  AD  5240  8F  3C 002FE     MOVZWL  #21056, FUNC_VAL                        ; 2657
                                                04  AE      01  D0 00304     MOVL    #1, YES_NO_ECHO                         ; 2658
                                                            06  11 00308     BRB     43$                                     ; 2653
                                                B4  AD  5200  8F  3C 0030A 42$:     MOVZWL  #20992, FUNC_VAL                ; 2661
                                                            5A  D5 00310 43$:     TSTL    PROT_OK                                 ; 2671
                                                            08  13 00312         BEQL    44$                                     ;
                                                    10  AE  D5 00314         TSTL    CONV_OK                                 ;
                                                            03  13 00317         BEQL    44$                                     ;
                                                          041C  31 00319         BRW     103$                                    ;
                                                            58  D4 0031C 44$:     CLRL    TERM_SEEN                               ; 2672
                                                    0C  AE  D5 0031E         TSTL    REPROMPT_DONE                           ; 2676
                                                            03  13 00321         BEQL    45$                                     ;
                                                          008D  31 00323         BRW     49$                                     ;
                                        03 00000000G        00  D1 00326 45$:     CMPL    COB$ACC_TERM_TYPE, #3                   ; 2688
                                                            55  12 0032D         BNEQ    47$                                     ;
                                                51      D0  AD  E9 0032F         BLBC    YES_PROTECT, 47$                        ; 2691
                                    FE68  CD    EC  AD      20  AE  28 00333     MOVC3   ON_LEN, ON_BUF, FIELD_BUF               ; 2706
                                                56      20  AE  D0 0033B         MOVL    ON_LEN, FIELD_LEN                       ; 2707
                AC  AD                  20          6E      00  2C 0033F     MOVC5   #0, (SP), #32, ACC_SIZE, FIELD_BUF-     ; 2708
                                                                FE68 CD46       00345   [FIELD_LEN]
                                                50      AC  AD  3C 00349     MOVZWL  ACC_SIZE, R0                            ; 2709
                                                56          50  C0 0034D     ADDL2   R0, FIELD_LEN
                AC  AD                  08          6E      00  2C 00350     MOVC5   #0, (SP), #8, ACC_SIZE, FIELD_BUF-      ; 2710
                                                                FE68 CD46       00356   [FIELD_LEN]
                                                50      AC  AD  3C 0035A     MOVZWL  ACC_SIZE, R0                            ; 2711
                                                56          50  C0 0035E     ADDL2   R0, FIELD_LEN
                                    000003F2    8F          56  D1 00361     CMPL    FIELD_LEN, #1010                        ; 2721
                                                            0D  15 00368     BLEQ    46$                                     ;
                                            00000000G       8F  DD 0036A     PUSHL   #COB$_ERRDURACC                         ; 2723
                                    00000000G   00          01  FB 00370     CALLS   #1, LIB$STOP
                                                      0240  8F  BB 00377 46$:     PUSHR   #^M<R6,R9>                              ; 2729
                                                     FE68   CD  9F 0037B         PUSHAB  FIELD_BUF
                                            0000V   CF          03  FB 0037F     CALLS   #3, COB$$RMS_PUT_BUFFER
                                                            20  AE  D5 00384 47$:     TSTL    ON_LEN                          ; 2749
                                                            13  13 00387         BEQL    48$                                     ;
                                                51      01      D0  AD  D1 00389     CMPL    YES_PROTECT, #1
                                                            0D  13 0038D         BEQL    48$                                     ;
                                                            59  DD 0038F         PUSHL   R9                                      ; 2751
                                                        24  AE  DD 00391         PUSHL   ON_LEN
                                                        EC  AD  9F 00394         PUSHAB  ON_BUF
                                            0000V   CF          03  FB 00397     CALLS   #3, COB$$RMS_PUT_BUFFER
                                                57      14      BE  D0 0039C 48$:     MOVL    @20(SP), RAB                    ; 2757
                                                        98  AD  DD 003A0         PUSHL   PUT_HERE+4                              ; 2759
                                                7E      AC  AD  3C 003A3     MOVZWL  ACC_SIZE, -(SP)                         ; 2758
                                                        B4  AD  DD 003A7         PUSHL   FUNC_VAL
                                                            57  DD 003AA         PUSHL   RAB
                                            0000V   CF          04  FB 003AC     CALLS   #4, COB$$RMS_GET
                                                            03  11 003B1         BRB     50$                                     ; 2676
                                                        0C  AE  D4 003B3 49$:     CLRL    REPROMPT_DONE                           ; 2763
                                                  B0    AD  22  A7  3C 003B6 50$:     MOVZWL  34(RAB), CHARS_READ             ; 2776
                                                  B8    AD  0E  A7  9A 003BB         MOVZBL  14(RAB), TERM_SIZE              ; 2777
                                                  BC    AD  0C  A7  9A 003C0         MOVZBL  12(RAB), TERM_LOC               ; 2778
                                    000181C8    8F      08  A7  D1 003C5         CMPL    8(RAB), #98760                  ; 2787
```

```
                        0E  12 003CD         BNEQ    51$                          ;
            58          01  D0 003CF         MOVL    #1, TERM_SEEN                ; 2790
                  04    AC  DD 003D2         PUSHL   UNIT                         ; 2791
                  94    AD  9F 003D5         PUSHAB  PARAMETERS                   ;
     0000V  CF          02  FB 003D8         CALLS   #2, COB$$PARTIAL_SEQ         ;
        7F  8F    0C    A7  91 003DD  51$:   CMPB    12(RAB), #127                ; 2798
                        0D  12 003E2         BNEQ    52$                          ;
                        59  DD 003E4         PUSHL   R9                           ; 2800
                  04    AC  DD 003E6         PUSHL   UNIT                         ;
                  94    AD  9F 003E9         PUSHAB  PARAMETERS                   ;
     0000V  CF          03  FB 003EC         CALLS   #3, COB$$DELETE_KEY          ;
            03    D0    AD  E8 003F1  52$:   BLBS    YES_PROTECT, 54$             ; 2825
                      00A8  31 003F5  53$:   BRW     63$                          ;
                  B0    AD  D5 003F8  54$:   TSTL    CHARS_READ                   ; 2826
                        F8  13 003FB         BEQL    53$                          ;
  000181B8  8F    08    A7  D1 003FD         CMPL    8(RAB), #98744               ; 2828
                        EE  12 00405         BNEQ    53$                          ;
                        58  D5 00407         TSTL    TERM_SEEN                    ; 2829
                        EA  12 00409         BNEQ    53$                          ;
                        53  D4 0040B         CLRL    NO_CHAR                      ; 2831
                        52  D4 0040D  55$:   CLRL    HAVE_TERM                    ;
            01          52  D1 0040F  56$:   CMPL    HAVE_TERM, #1                ; 2850
                        03  12 00412         BNEQ    57$                          ;
                      008C  31 00414         BRW     64$                          ;
                        53  D4 00417  57$:   CLRL    NO_CHAR                      ; 2853
        1C  AE  5240    8F  3C 00419         MOVZWL  #21056, FUNC_VAL_2           ; 2855
            57    14    BE  D0 0041F         MOVL    @20(SP), RAB                 ; 2857
                  A0    AD  9F 00423         PUSHAB  NEXT_CHAR                    ; 2858
                        01  DD 00426         PUSHL   #1                           ;
                  24    AE  DD 00428         PUSHL   FUNC_VAL_2                   ;
                        57  DD 0042B         PUSHL   RAB                          ;
     0000V  CF          04  FB 0042D         CALLS   #4, COB$$RMS_GET             ;
                  22    A7  B5 00432         TSTW    34(RAB)                      ; 2868
                        08  12 00435         BNEQ    58$                          ;
            53          01  D0 00437         MOVL    #1, NO_CHAR                  ; 2871
        A0  AD    0C    A7  90 0043A         MOVB    12(RAB), NEXT_CHAR           ; 2872
        B8  AD    0E    A7  9A 0043F  58$:   MOVZBL  14(RAB), TERM_SIZE           ; 2874
        BC  AD    0C    A7  9A 00444         MOVZBL  12(RAB), TERM_LOC            ; 2875
        C4  AD          01  D0 00449         MOVL    #1, TERM_IN_NEXT             ; 2876
  000181C8  8F    08    A7  D1 0044D         CMPL    8(RAB), #98760               ; 2877
                        0B  12 00455         BNEQ    59$                          ;
                  04    AC  DD 00457         PUSHL   UNIT                         ; 2879
                  94    AD  9F 0045A         PUSHAB  PARAMETERS                   ;
     0000V  CF          02  FB 0045D         CALLS   #2, COB$$PARTIAL_SEQ         ;
            2D          53  E9 00462  59$:   BLBC    NO_CHAR, 62$                 ; 2886
            5A          01  D0 00465         MOVL    #1, PROT_OK                  ; 2889
            52          01  D0 00468         MOVL    #1, HAVE_TERM                ; 2890
        7F  8F    0C    A7  91 0046B         CMPB    12(RAB), #127                ; 2901
                        9D  12 00470         BNEQ    56$                          ;
                        59  DD 00472         PUSHL   R9                           ; 2904
                  04    AC  DD 00474         PUSHL   UNIT                         ;
                  94    AD  9F 00477         PUSHAB  PARAMETERS                   ;
     0000V  CF          03  FB 0047A         CALLS   #3, COB$$DELETE_KEY          ;
                  B8    AD  D5 0047F         TSTL    TERM_SIZE                    ; 2910
                        06  12 00482         BNEQ    61$                          ;
                        52  D4 00484         CLRL    HAVE_TERM                    ; 2913
                        5A  D4 00486         CLRL    PROT_OK                      ; 2914
```

```
                                                85  11  00488 60$:     BRB     56$                                             ;2910
                                        52      01  D0  0048A 61$:     MOVL    #1, HAVE_TERM                                   ;2918
                                    C4          AD  D4  0048D           CLRL    TERM_IN_NEXT                                    ;2919
                                                F6  11  00490           BRB     60$                                             ;2901
                                                59  DD  00492 62$:     PUSHL   R9                                              ;2939
                                                02  DD  00494           PUSHL   #2                                              ;
                                    0000V       CF  02  FB  00496       CALLS   #2, COB$$RMS_PUT_BYTE                           ;
                                                5A  D4  0049B           CLRL    PROT_OK                                         ;2940
                                            FF6D    31  0049D           BRW     55$                                             ;2941
                                        5A          01  D0  004A0 63$:     MOVL    #1, PROT_OK                                     ;2956
                                        6B          5A  E9  004A3 64$:     BLBC    PROT_OK, 70$                                    ;2962
                                        07      C4  AD  E9  004A6           BLBC    TERM_IN_NEXT, 65$                               ;2969
                                    C0      AD      A0  AD  9E  004AA       MOVAB   NEXT_CHAR, TERM_PTR                             ;2971
                                                07  11  004AF           BRB     66$                                             ;
                                C0  AD      98  AD      B0  AD  C1  004B1 65$:     ADDL3   CHARS_READ, PUT_HERE+4, TERM_PTR        ;2973
                                            01  B8  AD  D1  004B8 66$:     CMPL    TERM_SIZE, #1                                   ;2990
                                                5D  12  004BC           BNEQ    72$                                             ;
                                    C0      AD  0C  A7  9E  004BE       MOVAB   12(RAB), TERM_PTR                               ;2993
                                            50  0C  A7  9A  004C3       MOVZBL  12(RAB), R0                                     ;2994
                                            09      50  91  004C7       CMPB    R0, #9                                          ;2996
                                                05  13  004CA           BEQL    67$                                             ;
                                            0D      50  91  004CC       CMPB    R0, #13                                         ;
                                                15  12  004CF           BNEQ    68$                                             ;
                                            06      6C  91  004D1 67$:     CMPB    (AP), #6                                        ;3001
                                                3B  1F  004D4           BLSSU   70$                                             ;
                                        18      AC  D5  004D6           TSTL    24(AP)                                          ;
                                                36  13  004D9           BEQL    70$                                             ;
                                            50  18      AC  D0  004DB       MOVL    KEY, R0                                         ;3003
                                    04      B0      C0  BD  90  004DF       MOVB    @TERM_PTR, @4(R0)                               ;
                                                2B  11  004E4           BRB     70$                                             ;3001
                                            1A      50  91  004E6 68$:     CMPB    R0, #26                                         ;3005
                                                13  12  004E9           BNEQ    69$                                             ;
                            24                  59  0B  E0  004EB       BBS     #11, R9, 71$                                    ;3013
                                                59  DD  004EF           PUSHL   R9                                              ;3028
                                        94      AD  9F  004F1           PUSHAB  PARAMETERS                                      ;
                                    0000V       CF  02  FB  004F4       CALLS   #2, COB$$CLEAN_UP                               ;
                                        18      AC  DD  004F9           PUSHL   KEY                                             ;3029
                                                40  11  004FC           BRB     73$                                             ;
                                        7F      8F      50  91  004FE 69$:     CMPB    R0, #127                                        ;3034
                                                0F  12  00502           BNEQ    71$                                             ;
                                                59  DD  00504           PUSHL   R9                                              ;3037
                                        04      AC  DD  00506           PUSHL   UNIT                                            ;
                                        94      AD  9F  00509           PUSHAB  PARAMETERS                                      ;
                                    0000V       CF  03  FB  0050C       CALLS   #3, COB$$DELETE_KEY                             ;
                                                64  11  00511 70$:     BRB     77$                                             ;2994
                                            CC      AD  D4  00513 71$:     CLRL    LEGAL                                           ;3043
                                        18      AC  DD  00516           PUSHL   KEY                                             ;3045
                                                4F  11  00519           BRB     76$                                             ;3044
                                            52      18  AC  D0  0051B 72$:     MOVL    KEY, R2                                         ;3062
                                        B0      AD  D5  0051F           TSTL    CHARS_READ                                      ;3051
                                                25  12  00522           BNEQ    74$                                             ;
                            0001827A        8F      08  A7  D1  00524       CMPL    8(RAB), #98938                                  ;
                                                1B  12  0052C           BNEQ    74$                                             ;
                            33                  59  0B  E0  0052E       BBS     #11, R9, 75$                                    ;3058
                                                59  DD  00532           PUSHL   R9                                              ;3066
                                        94      AD  9F  00534           PUSHAB  PARAMETERS                                      ;
                                    0000V       CF  02  FB  00537       CALLS   #2, COB$$CLEAN_UP                               ;
```

```
                                      52  DD 0053C          PUSHL   R2                                      ; 3067
                                  04  AC  DD 0053E 73$:     PUSHL   UNIT
                          0000V   CF      02  FB 00541     CALLS   #2, COB$$CONTROL_Z
                                        0226  31 00546     BRW     106$                                    ; 3068
                                    06    6C  91 00549 74$: CMPB    (AP), #6                                ; 3079
                                          17  1F 0054C     BLSSU   75$
                                  18  AC  D5 0054E         TSTL    24(AP)
                                          12  13 00551     BEQL    75$
                                          52  DD 00553     PUSHL   R2                                      ; 3087
                                  B8  AD  DD 00555         PUSHL   TERM_SIZE
                                  C0  AD  9F 00558         PUSHAB  TERM_PTR
                  00000000G       00      03  FB 0055B     CALLS   #3, COB$$CONTROL_KEY
                                    12    50  E8 00562     BLBS    R0, 77$
                                  CC  AD  D4 00565 75$:     CLRL    LEGAL                                   ; 3101
                                          52  DD 00568     PUSHL   R2                                      ; 3102
                                          59  DD 0056A 76$: PUSHL   R9
                                  04  AC  DD 0056C         PUSHL   UNIT
                                  94  AD  9F 0056F         PUSHAB  PARAMETERS
                          0000V   CF      04  FB 00572     CALLS   #4, COB$$ILLEGAL_TERM
                                          52  D4 00577 77$: CLRL    R2                                      ; 3123
                                  B0  AD  D5 00579         TSTL    CHARS_READ
                                          10  12 0057C     BNEQ    78$
                                          52  D6 0057E     INCL    R2
                      0A                  59  0B  E1 00580 BBC     #11, R9, 78$
                                          59  DD 00584     PUSHL   R9                                      ; 3130
                          0000V   CF      01  FB 00586     CALLS   #1, COB$$RPG_CLEAN_UP
                                        01DD  31 00588     BRW     105$                                    ; 3131
                                    45    52  E9 0058E 78$: BLBC    R2, 80$                                 ; 3134
                                    04    6C  91 00591     CMPB    (AP), #4                                ; 3136
                                          40  1F 00594     BLSSU   80$
                                  10  AC  D5 00596         TSTL    16(AP)
                                          3B  13 00599     BEQL    80$
                                  D4  AD  D5 00598         TSTL    YES_DEFAULT
                                          36  12 0059E     BNEQ    80$
                  B0  AD          10  BC  3C 005A0         MOVZWL  @DEFAULT, CHARS_READ                    ; 3140
                  D4  AD              01  D0 005A5         MOVL    #1, YES_DEFAULT                         ; 3141
                                    26    D0  AD E9 005A9 BLBC    YES_PROTECT, 79$                        ; 3149
                                    50    08  AC D0 005AD MOVL    STRING_DEST, R0                         ; 3150
                                    0A    02  A0 91 005B1 CMPB    2(R0), #10
                                          1C  13 005B5     BEQL    79$
                                    0B    02  A0 91 005B7 CMPB    2(R0), #11                              ; 3151
                                          16  13 005BB     BEQL    79$
                  AC  AD          10  BC  B1 005BD         CMPW    @DEFAULT, ACC_SIZE                      ; 3161
                                          0F  1B 005C2     BLEQU   79$
                              00000000G   8F  DD 005C4     PUSHL   #COB$_INVDEFVAL                         ; 3163
                  00000000G       00      01  FB 005CA     CALLS   #1, LIB$STOP
                                          03  11 005D1     BRB     80$
                                    5A    01  D0 005D3 79$: MOVL    #1, PROT_OK                             ; 3167
                                    5A    5A  E9 005D6 80$: BLBC    PROT_OK, 86$                            ; 3180
                                    20    08  AE E9 005D9 BLBC    YES_CONV, 81$                           ; 3182
                                          6E  DD 005DD     PUSHL   YES_SIGN                                ; 3186
                                  D4  AD  DD 005DF         PUSHL   YES_DEFAULT
                                  B0  AD  DD 005E2         PUSHL   CHARS_READ                              ; 3185
                                  94  AD  9F 005E5         PUSHAB  PUT_HERE                                ; 3184
                                  10  AC  DD 005E8         PUSHL   DEFAULT                                 ; 3185
                                          59  DD 005EB     PUSHL   R9                                      ; 3184
                                  08  AC  DD 005ED         PUSHL   STRING_DEST
```

```
                    00000000G  00              07  FB  005F0          CALLS   #7, COB$$ACC_CONVERT
                            10  AE              50  D0  005F7          MOVL    R0, CONV_OK
                                                36  11  005FB          BRB     86$
     B0  AD      08  BC                 10      00  ED  005FD  81$:     CMPZV   #0, #16, @STRING_DEST, CHARS_READ                3196
                                                07  15  00604          BLEQ    82$
                            2C  AE          B0  AD  D0  00606          MOVL    CHARS_READ, COPY_NUM                             3198
                                                05  11  0060B          BRB     83$
                            2C  AE          08  BC  3C  0060D  82$:     MOVZWL  @STRING_DEST, COPY_NUM                           3200
                            09  D4          AD  E9  00612  83$:     BLBC    YES_DEFAULT, 84$                                 3203
                            50  10          AC  D0  00616          MOVL    DEFAULT, R0                                      3204
                                04          A0  DD  0061A          PUSHL   4(R0)
                                            03  11  0061D          BRB     85$
                                98          AD  DD  0061F  84$:     PUSHL   PUT_HERE+4                                       3205
                                30          AE  9F  00622  85$:     PUSHAB  COPY_NUM                                         3202
                                08          AC  DD  00625          PUSHL   STRING_DEST
                    00000000G  00              03  FB  00628          CALLS   #3, STR$COPY_R
                            10  AE              01  D0  0062F          MOVL    #1, CONV_OK                                      3208
                                            10  AE  D5  00633  86$:     TSTL    CONV_OK                                          3215
                                                03  13  00636          BEQL    87$
                                            00F6  31  00638          BRW     101$
                12                  59          08  E1  0063B  87$:     BBC     #11, R9, 88$                                     3229
                                                59  DD  0063F          PUSHL   R9                                               3236
                                                02  DD  00641          PUSHL   #2
                        0000V   CF              02  FB  00643          CALLS   #2, COB$$RMS_PUT_BYTE
                                                59  DD  00648          PUSHL   R9                                               3237
                        0000V   CF              01  FB  0064A          CALLS   #1, COB$$RPG_CLEAN_UP
                                                10  11  0064F          BRB     89$                                              3238
                                02          05  AC  91  00651  88$:     CMPB    UNIT+1, #2                                       3241
                                                0D  12  00655          BNEQ    90$
                                                59  DD  00657          PUSHL   R9                                               3247
                                94          AD  9F  00659          PUSHAB  PARAMETERS
                        0000V   CF              02  FB  0065C          CALLS   #2, COB$$CLEAN_UP
                                            010B  31  00661  89$:     BRW     106$                                             3248
                                04          18  AE  E8  00664  90$:     BLBS    24(SP), 91$                                      3258
                                0C          D4  AD  E9  00668          BLBC    YES_DEFAULT, 92$
                                                59  DD  0066C  91$:     PUSHL   R9                                               3261
                                                02  DD  0066E          PUSHL   #2
                        0000V   CF              02  FB  00670          CALLS   #2, COB$$RMS_PUT_BYTE
                                            00B1  31  00675          BRW     100$                                             3258
                                                56  D4  00678  92$:     CLRL    PUT_TOTAL                                        3264
                                                58  D4  0067A          CLRL    INDEX
                                                59  DD  0067C          PUSHL   R9
                                                02  DD  0067E          PUSHL   #2
                        0000V   CF              02  FB  00680          CALLS   #2, COB$$RMS_PUT_BYTE
                                04          AE  D5  00685          TSTL    YES_NO_ECHO
                                                59  12  00688          BNEQ    96$
                                                5B  D4  0068A          CLRL    R11
                                D8          AD  D5  0068C          TSTL    PUT_FLAG
                                                12  13  0068F          BEQL    93$
                                                5B  D6  00691          INCL    R11
                                D0          AD  D5  00693          TSTL    YES_PROTECT
                                                0B  12  00696          BNEQ    93$
 30  AE      DC  AD          E8  AD  28  00698          MOVC3   OFF_LEN, OFF_BUF, RESTORE_CURSOR
                        56      E8  AD  D0  0069F          MOVL    OFF_LEN, PUT_TOTAL
                        58              56  D0  006A3  93$:     MOVL    PUT_TOTAL, INDEX
      51                56      B0  AD  C1  006A6          ADDL3   CHARS_READ, PUT_TOTAL, R1
                        50      FF  A6  9E  006AB          MOVAB   -1(R6), P
```

```
                                        12  11 006AF          BRB     95$                                              ;
               30 AE48                  08  90 006B1 94$:     MOVB    #8, RESTORE_CURSOR[INDEX]                        ;
               31 AE48                  20  90 006B6          MOVB    #32, RESTORE_CURSOR+1[INDEX]                     ;
               32 AE48                  08  90 006BB          MOVB    #8, RESTORE_CURSOR+2[INDEX]                      ;
                     58                 03  C0 006C0          ADDL2   #3, INDEX                                        ;
        EA           50                 51  F2 006C3 95$:     AOBLSS  R1, P, 94$                                       ;
        50      B0   AD                 03  C5 006C7          MULL3   #3, CHARS_READ, R0                               ;
                     56                 50  C0 006CC          ADDL2   R0, PUT_TOTAL                                    ;
                     11                 5B  E9 006CF          BLBC    R11, 96$                                         ;
                               D0       AD  D5 006D2          TSTL    YES_PROTECT                                      ;
                                        0C  12 006D5          BNEQ    96$                                              ;
     30 AE46    EC   AD        20       AE  28 006D7          MOVC3   ON_LEN, ON_BUF, RESTORE_CURSOR[PUT_TOTAL]        ;
                     56        20       AE  C0 006DF          ADDL2   ON_LEN, PUT_TOTAL                                ;
                                        51  D4 006E3 96$:     CLRL    LAST_WRITE                                       ;
                                        51  D5 006E5 97$:     TSTL    LAST_WRITE                                       ;
                                        18  12 006E7          BNEQ    99$                                              ;
          000003F2   8F                 56  D1 006E9          CMPL    PUT_TOTAL, #1010                                 ;
                                        0A  15 006F0          BLEQ    98$                                              ;
                     50      03F2       8F  3C 006F2          MOVZWL  #1010, P_TOT                                     ;
                     56                 50  C2 006F7          SUBL2   P_TOT, PUT_TOTAL                                 ;
                                        E9  11 006FA          BRB     97$                                              ;
                     50                 56  D0 006FC 98$:     MOVL    PUT_TOTAL, P_TOT                                 ;
                     51                 01  D0 006FF          MOVL    #1, LAST_WRITE                                   ;
                                        E1  11 00702          BRB     97$                                              ;
                             0201       8F  BB 00704 99$:     PUSHR   #^M<R0,R9>                                       ;
                               38       AE  9F 00708          PUSHAB  RESTORE_CURSOR                                   ;
             0000V   CF                 03  FB 0070B          CALLS   #3, COB$$RMS_PUT_BUFFER                          ;
                     57        14       BE  D0 00710          MOVL    @20(SP), RAB                                     ;
                               98       AD  DD 00714          PUSHL   PUT_HERE+4                                       ;
                     7E        AC       AD  3C 00717          MOVZWL  ACC_SIZE, -(SP)                                  ;
                               B4       AD  DD 0071B          PUSHL   FUNC_VAL                                         ;
                                        57  DD 0071E          PUSHL   RAB                                              ;
             0000V   CF                 04  FB 00720          CALLS   #4, COB$$RMS_GET                                 ;
               0C    AE                 01  D0 00725          MOVL    #1, REPROMPT_DONE                                ;
                               10       AE  D4 00729 100$:    CLRL    CONV_OK                                          ; 3267
                               D4       AD  D4 0072C          CLRL    YES_DEFAULT                                      ; 3268
                                        04  11 0072F          BRB     102$                                             ; 3215
               10    AE                 01  D0 00731 101$:    MOVL    #1, CONV_OK                                      ; 3277
                             FBD8       31 00735 102$:        BRW     43$                                              ; 2671
               07                       6C  91 00738 103$:    CMPB    (AP), #7                                         ; 3286
                                        0A  1F 0073B          BLSSU   104$                                             ;
                               1C       AC  D5 0073D          TSTL    28(AP)                                           ;
                                        05  13 00740          BEQL    104$                                             ;
               1C    BC        B0       AD  D0 00742          MOVL    CHARS_READ, @LENGTH                              ; 3288
                                        59  DD 00747 104$:    PUSHL   R9                                               ; 3299
                               94       AD  9F 00749          PUSHAB  PARAMETERS                                       ;
             0000V   CF                 02  FB 0074C          CALLS   #2, COB$$CLEAN_UP                                ;
                               94       AD  9F 00751          PUSHAB  PUT_HERE                                         ; 3307
          00000000G  00                 01  FB 00754          CALLS   #1, STR$FREE1_DX                                 ;
                     0D                 50  E8 00758          BLBS    R0, 105$                                         ;
                        00000000G       8F  DD 0075E          PUSHL   #COB$_ERRDURACC                                  ; 3308
          00000000G  00                 01  FB 00764          CALLS   #1, LIB$STOP                                     ;
                     50                 01  D0 0076B 105$:    MOVL    #1, R0                                           ; 3311
                                            04 0076E          RET                                                      ;
                                        50  D4 0076F 106$:    CLRL    R0                                               ; 3312
                                            04 00771          RET                                                      ;
```

; Routine Size: 1906 bytes,   Routine Base: _COB$CODE + 0238

```
;  1801          3313  1 %SBTTL 'COB$ACC_SCR_FILE - Screen enhancements for files'
;  1802          3314  1 ROUTINE COB$$ACC_SCR_FILE ( UNIT          :  VECTOR [2,BYTE],
;  1803          3315  1                             STRING_DEST   :  REF $STR$DESCRIPTOR,
;  1804          3316  1                             FLAGS,
;  1805          3317  1                             DEFAULT       :  REF $STR$DESCRIPTOR,
;  1806          3318  1                             LENGTH,
;  1807          3319  1                             ACC_SIZE,
;  1808          3320  1                             PUT_HERE      :  REF BLOCK [8, BYTE],
;  1809          3321  1                                                 ! Contains input characters
;  1810          3322  1                             YES_CONV,           ! =1 if conversion requested
;  1811          3323  1                             YES_PROTECT,        ! =1 if protection requested
;  1812          3324  1                             YES_SIGN            ! =1 if sign should be included
;  1813          3325  1                           ) =
;  1814          3326  1
;  1815          3327  1 !++
;  1816          3328  1 ! FUNCTIONAL DESCRIPTION:
;  1817          3329  1 !
;  1818          3330  1 !       This routine handles the VAX COBOL Version 3 ACCEPT statement
;  1819          3331  1 !       with Screen Enhancements when a file (not a terminal) is used
;  1820          3332  1 !       for input.  A non terminal $GET service does not contain all the
;  1821          3333  1 !       features of a terminal $GET service, so this routine is a scaled
;  1822          3334  1 !       down version of COB$ACC_SCR.  Note that the fields RAB [RAB$V_ETO]
;  1823          3335  1 !       and RAB [RAB$L_XAB] are not set.
;  1824          3336  1 !
;  1825          3337  1 ! FORMAL PARAMETERS:
;  1826          3338  1 !
;  1827          3339  1 !       UNIT.rbu.va     Array of two unsigned byte integers.
;  1828          3340  1 !                       The first byte is the unit number designating the
;  1829          3341  1 !                       device from which the string is to be read.
;  1830          3342  1 !                       The second byte indicates whether the routine should
;  1831          3343  1 !                       abort or return to the calling program.
;  1832          3344  1 !                         Byte 2 = 0  -  routine will abort on control z
;  1833          3345  1 !                                        and reprompt on conversion errors.
;  1834          3346  1 !                                = 1  -  ( AT END )
;  1835          3347  1 !                                        routine will return to calling program
;  1836          3348  1 !                                        on control z and reprompt on conversion
;  1837          3349  1 !                                        errors.
;  1838          3350  1 !                                = 2  -  ( ON EXCEPTION )
;  1839          3351  1 !                                        routine will return to calling program
;  1840          3352  1 !                                        on control z and conversion errors.
;  1841          3353  1 !
;  1842          3354  1 !       STRING_DEST.mt.ds   Address of descriptor to receive the read input.
;  1843          3355  1 !
;  1844          3356  1 !       FLAGS.rlu.v     Screen enhancement flag;
;  1845          3357  1 !
;  1846          3358  1 !                               bit 0  -  bold
;  1847          3359  1 !                               bit 1  -  reverse
;  1848          3360  1 !                               bit 2  -  blink
;  1849          3361  1 !                               bit 3  -  underline
;  1850          3362  1 !                               bit 4  -  bell
;  1851          3363  1 !                               bit 5  -  conversion
;  1852          3364  1 !                               bit 6  -  decimal point is comma
;  1853          3365  1 !                               bit 7  -  0 to allow space for sign in PROTECTED
;  1854          3366  1 !                                         ACCEPT, 1 no allowance for sign
;  1855          3367  1 !                               bit 8  -  protect
;  1856          3368  1 !                               bit 9  -  no-echo
;  1857          3369  1 !                               bit 10 -  0 advancing, 1 no advancing
```

```
;  1858          3370  1 !                                   bit 11 -  0 for VAX COBOL, 1 for VAX RPG
;  1859          3371  1 !
;  1860          3372  1 !       DEFAULT.rt.dx   Default source moved to destination descriptor
;  1861          3373  1 !                       (STRING_DEST) in the event of null input.
;  1862          3374  1 !
;  1863          3375  1 !       LENGTH.wlu.r    Destination of the number of characters read.
;  1864          3376  1 !
;  1865          3377  1 !       ACC_SIZE.rlu.v  # of characters to RMS $GET.
;  1866          3378  1 !
;  1867          3379  1 !       PUT_HERE.rt.dx  Buffer to hold input characters.
;  1868          3380  1 !
;  1869          3381  1 !       YES_CONV.rlu.v  Flag = 1 if Conversion requested by user.
;  1870          3382  1 !
;  1871          3383  1 !       YES_PROTECT.rlu.v  Flag = 1 if Protection requested by user.
;  1872          3384  1 !
;  1873          3385  1 !       YES_SIGN.rlu.v  Flag = 1 if sign should be included in COMP or COMP3
;  1874          3386  1 !                       data type.
;  1875          3387  1 !
;  1876          3388  1 ! IMPLICIT INPUTS:
;  1877          3389  1 !
;  1878          3390  1 !       Status of whether the input file is currently open.
;  1879          3391  1 !
;  1880          3392  1 ! IMPLICIT OUTPUTS:
;  1881          3393  1 !
;  1882          3394  1 !       Updated status of file
;  1883          3395  1 !
;  1884          3396  1 ! ROUTINE VALUE:
;  1885          3397  1 !
;  1886          3398  1 !       If .UNIT[1] is false :  Unspecified.
;  1887          3399  1 !       If .UNIT[1] is true  :  Either true or false, indicating success or
;  1888          3400  1 !                               EOF, respectively.
;  1889          3401  1 !
;  1890          3402  1 ! SIDE EFFECTS:
;  1891          3403  1 !
;  1892          3404  1 !       Reads a record from a designated uint.
;  1893          3405  1 !
;  1894          3406  1 !--
;  1895          3407  1
;  1896          3408  2     BEGIN
;  1897          3409  2
;  1898          3410  2     LOCAL
;  1899          3411  2         RAB             :  REF $RAB_DECL,
;  1900          3412  2         CR_BUF          :  VECTOR [LT,BYTE],
;  1901          3413  2         CHARS_READ      :  INITIAL (0),         ! Number of characters read
;  1902          3414  2         CONV_OK         :  INITIAL (0),         ! = 1 if no conversion errors
;  1903          3415  2         YES_DEFAULT     :  INITIAL (0) ;        ! = 1 if DEFAULT was used as input
;  1904          3416  2
;  1905          3417  2     BUILTIN
;  1906          3418  2         NULLPARAMETER ;
;  1907          3419  2
;  1908          3420  2     !+
;  1909          3421  2     !  RMS $PUT - If previous call requires advancing, $PUT a linefeed to
;  1910          3422  2     !  SYS$OUTPUT.  Open SYS$OUTPUT if necessary.
;  1911          3423  2     !-
;  1912          3424  2
;  1913          3425  3     IF (.COB$$AB_PREV[0] EQL DISP
;  1914          3426  3         OR .COB$$AB_PREV[0] EQL POS
```

```
;  1915          3427  3          OR .COB$$AB_PREV[0] EQL ACC_ADV )
;  1916          3428  3      THEN
;  1917          3429  3          COB$$RMS_PUT_BYTE ( LINE_FD, .FLAGS ) ;
;  1918          3430
;  1919          3431  2      !+
;  1920          3432  2      !  RMS $GET to accept input from a file.
;  1921          3433  2      !-
;  1922          3434
;  1923          3435  2      RAB = .COB$$AL_WRITE_RAB [.UNIT[0]] ;
;  1924          3436  2      RAB [RAB$W_USZ] = .ACC_SIZE ;
;  1925          3437  2      RAB [RAB$L_UBF] = .PUT_HERE [DSC$A_POINTER] ;
;  1926          3438  2      !+
;  1927          3439  2      !  Turn off RAB [RAB$V_ETO] just in case a 'screen enhancement ACCEPT'
;  1928          3440  2      !  was performed before this one.
;  1929          3441  2      !-
;  1930          3442  2      RAB [RAB$V_ETO] = 0 ;
;  1931          3443  2
;  1932          3444  2      WHILE $GET (RAB = .RAB) EQL RMS$_RSA DO $WAIT (RAB = .RAB) ;
;  1933          3445  2
;  1934          3446  2      IF NOT .RAB [RAB$L_STS] AND NULLPARAMETER (DEFAULT)
;  1935          3447  2      THEN
;  1936          3448  2          LIB$STOP (( IF .RAB[RAB$L_STS] EQL RMS$_EOF
;  1937          3449  2                      THEN
;  1938          3450  2                          !+
;  1939          3451  2                          !  If ON EXCEPTION or AT END, return to user program.
;  1940          3452  2                          !-
;  1941          3453  2                          IF .UNIT [1] EQL 1 OR .UNIT [1] EQL 2
;  1942          3454  2                          THEN
;  1943          3455  2                              RETURN 0
;  1944          3456  2                          ELSE
;  1945          3457  2                              COB$_EOFON_ACC
;  1946          3458  2                      ELSE
;  1947          3459  2                          COB$_ERRDURACC),
;  1948          3460  2                  1, .RAB + RAB$C_BLN, .RAB [RAB$L_STS], .RAB [RAB$L_STV] ) ;
;  1949          3461  2
;  1950          3462  2      !+
;  1951          3463  2      !  Put number of characters read from $GET in CHARS_READ.
;  1952          3464  2      !  Pass this info along to COB$$ACC_CONVERT.
;  1953          3465  2      !-
;  1954          3466  2
;  1955          3467  2      CHARS_READ = .RAB [RAB$W_RSZ] ;                     ! Number of chars read
;  1956          3468  2
;  1957          3469  2  !*******
;  1958          3470  2  !******* NULL INPUT
;  1959          3471  2  !*******
;  1960          3472  2
;  1961          3473  2      !+
;  1962          3474  2      !  Null input.
;  1963          3475  2      !  Check for DEFAULT parameter - if present prepare to put it through
;  1964          3476  2      !  Conversion routines by placing DEFAULT in PUT_HERE.
;  1965          3477  2      !-
;  1966          3478  2
;  1967          3479  2      IF ( .CHARS_READ EQL 0 ) AND (( .FLAGS AND V_COB_RPG ) NEQ 0 )
;  1968          3480  2      THEN
;  1969          3481  2          !+
;  1970          3482  2          !  In case of null input for RPG, simply return (no DEFAULT),
;  1971          3483  2          !  after setting advancing flag.
```

```
;  1972     3484  2          !-
;  1973     3485  2          BEGIN
;  1974     3486  2          IF (.FLAGS AND V_ADV) NEQ 0
;  1975     3487  2          THEN
;  1976     3488  2              COB$$AB_PREV[0] = ACC_DNA
;  1977     3489  2          ELSE
;  1978     3490  2              COB$$AB_PREV[0] = ACC_ADV ;
;  1979     3491  2          RETURN 1 ;
;  1980     3492  2          END ;
;  1981     3493  2
;  1982     3494  2      !+
;  1983     3495  2      ! There can be no PROTECTION check on input when dealing with files as
;  1984     3496  2      ! RMS will only read ACC_SIZE characters or less.  If .ACC_SIZE were 4
;  1985     3497  2      ! but the record contained "abcdef", only "abcd" will be pulled from the
;  1986     3498  2      ! record.  RMS ignores the remaining characters "ef" and goes on to the
;  1987     3499  2      ! next record.  However it is possible to perform a PROTECTION check
;  1988     3500  2      ! when the DEFAULT value is used.
;  1989     3501  2      !-
;  1990     3502  2
;  1991     3503  2      IF ( .CHARS_READ EQL 0 )
;  1992     3504  2      THEN
;  1993     3505  2          BEGIN
;  1994     3506  2          IF (.DEFAULT NEQ 0) AND (.YES_DEFAULT EQL 0)
;  1995     3507  2          THEN
;  1996     3508  2              BEGIN                                   ! Begin YES Default
;  1997     3509  2
;  1998     3510  2              CHARS_READ = .DEFAULT [DSC$W_LENGTH];
;  1999     3511  2              YES_DEFAULT = 1 ;
;  2000     3512  2
;  2001     3513  4              !+
;  2002     3514  4              ! Protection check for DEFAULT excluding the Floating
;  2003     3515  4              ! Point data types ( these will be handled in
;  2004     3516  4              ! COB$$VERIFY_FL_RANGE.
;  2005     3517  4              !-
;  2006     3518  4
;  2007     3519  5              IF (.YES_PROTECT AND
;  2008     3520  6                      ( .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_F AND
;  2009     3521  5                        .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_D ))
;  2010     3522  4              THEN                                    ! Check protection
;  2011     3523  5                  IF (.DEFAULT [DSC$W_LENGTH] GTR .ACC_SIZE)
;  2012     3524  4                  THEN
;  2013     3525  4                      !+
;  2014     3526  4                      ! If the length of DEFAULT is greater than the
;  2015     3527  4                      ! expected input size ACC_SIZE, then there is a
;  2016     3528  4                      ! Protection error.
;  2017     3529  4                      !-
;  2018     3530  4                      LIB$STOP ( COB$_INVDEFVAL ) ;
;  2019     3531  4
;  2020     3532  3              END ;                                   ! End YES Default
;  2021     3533  2          END ;
;  2022     3534  2
;  2023     3535  2  !*******
;  2024     3536  2  !******* CONVERSION
;  2025     3537  2  !*******
;  2026     3538  2
;  2027     3539  2      !+
;  2028     3540  2      ! If conversion requested, call routine COB$$ACC_CONVERT
```

```
;  2029          3541  2     !-
;  2030          3542  2
;  2031          3543  3     IF ( .YES_CONV )
;  2032          3544  3     THEN
;  2033          3545  3         CONV_OK = COB$$ACC_CONVERT ( .STRING_DEST, .FLAGS,
;  2034          3546  3                                      .DEFAULT, .PUT_HERE, .CHARS_READ,
;  2035          3547  3                                                .YES_DEFAULT, .YES_SIGN )
;  2036          3548  3     ELSE
;  2037          3549  3         BEGIN
;  2038          3550  3             LOCAL
;  2039          3551  3                 COPY_NUM ;
;  2040          3552  3
;  2041          3553  3             !+
;  2042          3554  3             ! No conversion requested - copy input data to STRING_DEST.
;  2043          3555  3             ! Use STR$COPY_R because it BLANK fills.
;  2044          3556  3             !-
;  2045          3557  3
;  2046          3558  3             IF .CHARS_READ LSS .STRING_DEST[DSC$W_LENGTH]
;  2047          3559  3             THEN
;  2048          3560  3                 COPY_NUM = .CHARS_READ
;  2049          3561  3             ELSE
;  2050          3562  3                 COPY_NUM = .STRING_DEST[DSC$W_LENGTH] ;
;  2051          3563  3
;  2052          3564  3             STR$COPY_R ( .STRING_DEST, COPY_NUM,
;  2053          3565  4                                        (IF .YES_DEFAULT
;  2054          3566  4                                        THEN .DEFAULT [DSC$A_POINTER]
;  2055          3567  3                                        ELSE .PUT_HERE [DSC$A_POINTER] )) ;
;  2056          3568  3
;  2057          3569  3             CONV_OK = 1 ;                       ! set CONV_OK to success
;  2058          3570  2             END;
;  2059          3571  2
;  2060          3572  2     !+
;  2061          3573  2     ! Conversion completed - was it successful ?
;  2062          3574  2     !-
;  2063          3575  2
;  2064          3576  2     IF .CONV_OK  EQL 0
;  2065          3577  2     THEN
;  2066          3578  2         !+
;  2067          3579  2         ! CONVERSION error.  Read UNIT parameter to determine what
;  2068          3580  2         ! to do.  There is no Reprompting done with Files as input.
;  2069          3581  2         !
;  2070          3582  2         !       Byte 2 of                       Conversion
;  2071          3583  2         !         UNIT                            error
;  2072          3584  2         !
;  2073          3585  2         !          0                              COB$_ERRDURACC
;  2074          3586  2         !          1 ( at end )                   COB$_ERRDURACC
;  2075          3587  2         !          2 ( on exception )             Return
;  2076          3588  2         !-
;  2077          3589  2
;  2078          3590  3         BEGIN                                   ! Begin conversion error
;  2079          3591  3         IF ( .FLAGS AND V_COB_RPG ) NEQ 0
;  2080          3592  3         THEN
;  2081          3593  3             !+
;  2082          3594  3             ! VAX RPG - return on a Conversion Error, ring bell
;  2083          3595  3             ! and clean up first.
;  2084          3596  3             !-
;  2085          3597  4             BEGIN
```

```
; 2086          3598  4               COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
; 2087          3599  4               COB$$RPG_CLEAN_UP ( .FLAGS );
; 2088          3600  4               RETURN 0 ;
; 2089          3601  3               END ;
; 2090          3602  3
; 2091          3603  3           IF .UNIT [1] EQL 2
; 2092          3604  3           THEN
; 2093          3605  3               RETURN 0
; 2094          3606  3           ELSE
; 2095          3607  3
; 2096          3608  3               !+
; 2097          3609  3               ! When dealing with a file, it was decided to return a fatal
; 2098          3610  3               ! error message rather then REPROMPT.  This lets the user know
; 2099          3611  3               ! where the problem in the file is so that the input file can
; 2100          3612  3               ! be corrected before running the program again.  Otherwise, the
; 2101          3613  3               ! the user might run out of data by the end of the program or the
; 2102          3614  3               ! reprompting process may lead to further conversion errors.
; 2103          3615  3               !-
; 2104          3616  3
; 2105          3617  3               LIB$STOP ( COB$_ERRDURACC ) ;
; 2106          3618  2           END ;                                   ! End conversion error
; 2107          3619  2
; 2108          3620  2       !+
; 2109          3621  2       ! Fill in optional LENGTH parameter with the number of
; 2110          3622  2       ! characters read if no error.
; 2111          3623  2       !-
; 2112          3624  2
; 2113          3625  2       IF .LENGTH NEQ 0
; 2114          3626  2       THEN
; 2115          3627  2           .LENGTH = .CHARS_READ ;
; 2116          3628  2
; 2117          3629  2   !*******
; 2118          3630  2   !******* CLEAN UP
; 2119          3631  2   !*******
; 2120          3632  2
; 2121          3633  2       !+
; 2122          3634  2       ! Determine if ADVANCING is requested.
; 2123          3635  2       ! If bit 10 = 0 advancing.  If bit 10 = 1 no advancing.
; 2124          3636  2       ! Set COB$$AB_PREV[0] - also depending on bit 10, to flag to next COBOL
; 2125          3637  2       ! statement that advancing/no advancing is required following this
; 2126          3638  2       ! ACCEPT statement.  Echo carriage return to screen if advancing is
; 2127          3639  2       ! called for.
; 2128          3640  2       !-
; 2129          3641  2
; 2130          3642  2       IF (.FLAGS AND V_ADV) NEQ 0
; 2131          3643  2       THEN
; 2132          3644  2           COB$$AB_PREV[0] = ACC_DNA                       ! No Advancing
; 2133          3645  2       ELSE
; 2134          3646  2           COB$$RMS_PUT_BYTE ( CARR_RET, .FLAGS ) ;        ! Advance via a carriage
; 2135          3647  2                                                           ! return
; 2136          3648  2       RETURN 1;
; 2137          3649  1       END;                                        ! End of COB$$ACC_SCR_FILE
```

```
                                         07FC 00000 COB$$ACC_SCR_FILE:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10             ; 3314
                5A       0000V  CF   9E 00002              MOVAB   COB$$RMS_PUT_BYTE, R10                       ;
                59 00000000G    8F   D0 00007              MOVL    #COB$_ERRDURACC, R9                          ;
                58 00000000G    00   9E 0000E              MOVAB   COB$$AB_PREV, R8                             ;
                57 00000000G    00   9E 00015              MOVAB   LIB$STOP, R7                                 ;
                5E              04   C2 0001C              SUBL2   #4, SP                                       ;
                                55   7C 0001F              CLRQ    CHARS_READ                                   ; 3408
                                54   D4 00021              CLRL    YES_DEFAULT                                  ;
                50              68   9A 00023              MOVZBL  COB$$AB_PREV, R0                             ; 3425
                                0A   13 00026              BEQL    1$                                           ;
                02              50   91 00028              CMPB    R0, #2                                       ; 3426
                                05   13 0002B              BEQL    1$                                           ;
                04              50   91 0002D              CMPB    R0, #4                                       ; 3427
                                08   12 00030              BNEQ    2$                                           ;
                         0C     AC   DD 00032 1$:          PUSHL   FLAGS                                        ; 3429
                                01   DD 00035              PUSHL   #1                                           ;
                6A              02   FB 00037              CALLS   #2, COB$$RMS_PUT_BYTE                        ;
                50       04     AC   9A 0003A 2$:          MOVZBL  UNIT, R0                                     ; 3435
                52 00000000G0040     D0 0003E              MOVL    COB$$AL_WRITE_RAB[R0], RAB                   ;
          20    A2       18     AC   B0 00046              MOVW    ACC_SIZE, 32(RAB)                            ; 3436
                53       1C     AC   D0 0004B              MOVL    PUT_HERE, R3                                 ; 3437
          24    A2       04     A3   D0 0004F              MOVL    4(R3), 36(RAB)                               ;
          07    A2              10   8A 00054              BICB2   #16, 7(RAB)                                  ; 3442
                                52   DD 00058 3$:          PUSHL   RAB                                          ; 3444
     00000000G  00              01   FB 0005A              CALLS   #1, SYS$GET                                  ;
     000182DA   8F              50   D1 00061              CMPL    R0, #99034                                   ;
                                0B   12 00068              BNEQ    4$                                           ;
                                52   DD 0006A              PUSHL   RAB                                          ;
     00000000G  00              01   FB 0006C              CALLS   #1, SYS$WAIT                                 ;
                                E3   11 00073              BRB     3$                                           ;
                3C       08     A2   E8 00075 4$:          BLBS    8(RAB), 10$                                  ; 3446
                04              6C   91 00079              CMPB    (AP), #4                                     ;
                                05   1F 0007C              BLSSU   5$                                           ;
                         10     AC   D5 0007E              TSTL    16(AP)                                       ;
                                32   12 00081              BNEQ    10$                                          ;
                7E       08     A2   7D 00083 5$:          MOVQ    8(RAB), -(SP)                                ; 3460
                         44     A2   9F 00087              PUSHAB  68(RAB)                                      ;
                                01   DD 0008A              PUSHL   #1                                           ; 3448
     0001827A   8F       08     A2   D1 0008C              CMPL    8(RAB), #98938                               ;
                                1A   12 00094              BNEQ    8$                                           ;
                01       05     AC   91 00096              CMPB    UNIT+1, #1                                   ; 3453
                                04   13 0009A              BEQL    6$                                           ;
                02       05     AC   91 0009C              CMPB    UNIT+1, #2                                   ;
                                03   12 000A0 6$:          BNEQ    7$                                           ;
                              00F9   31 000A2              BRW     26$                                          ;
                50 00000000G    8F   D0 000A5 7$:          MOVL    #COB$_EOFON_ACC, R0                          ;
                                50   DD 000AC              PUSHL   R0                                           ;
                                02   11 000AE              BRB     9$                                           ;
                                59   DD 000B0 8$:          PUSHL   R9                                           ; 3448
                67              05   FB 000B2 9$:          CALLS   #5, LIB$STOP                                 ;
                55       22     A2   3C 000B5 10$:         MOVZWL  34(RAB), CHARS_READ                          ; 3467
                                50   D4 000B9              CLRL    R0                                           ; 3479
                                55   D5 000BB              TSTL    CHARS_READ                                   ;
                                15   12 000BD              BNEQ    12$                                          ;
                                50   D6 000BF              INCL    R0                                           ;
       0E       0C      AC      0B   E1 000C1              BBC     #11, FLAGS, 12$                              ;
```

```
                03      0C   AC              0A  E1 000C6          BBC     #10, FLAGS, 11$                          ; 3486
                                          00BF  31 000CB          BRW     23$                                      ;
                                      68        04  90 000CE 11$:  MOVB    #4, COB$$AB_PREV                         ; 3490
                                          00C6  31 000D1          BRW     25$                                      ; 3491
                                      36        50  E9 000D4 12$:  BLBC    R0, 13$                                  ; 3503
                                            10  AC  D5 000D7          TSTL    DEFAULT                                  ; 3506
                                                31  13 000DA          BEQL    13$                                      ;
                                                54  D5 000DC          TSTL    YES_DEFAULT                              ;
                                                2D  12 000DE          BNEQ    13$                                      ;
                                      55        10  BC  3C 000E0          MOVZWL  @DEFAULT, CHARS_READ             ; 3510
                                      54        01  D0 000E4          MOVL    #1, YES_DEFAULT                          ; 3511
                                      22        24  AC  E9 000E7          BLBC    YES_PROTECT, 13$                 ; 3519
                                      50        08  AC  D0 000EB          MOVL    STRING_DEST, R0                  ; 3520
                                      0A        02  A0  91 000EF          CMPB    2(R0), #10                       ;
                                                18  13 000F3          BEQL    13$                                      ;
                                      0B        02  A0  91 000F5          CMPB    2(R0), #11                       ; 3521
                                                12  13 000F9          BEQL    13$                                      ;
    18  AC      10  BC              10              00  ED 000FB          CMPZV   #0, #16, @DEFAULT, ACC_SIZE      ; 3523
                                                09  15 00102          BLEQ    13$                                      ;
                                  00000000G     8F  DD 00104          PUSHL   #COB$_INVDEFVAL                  ; 3530
                                      67        01  FB 0010A          CALLS   #1, LIB$STOP                             ;
                                      1A        20  AC  E9 0010D 13$:  BLBC    YES_CONV, 14$                    ; 3543
                                                28  AC  DD 00111          PUSHL   YES_SIGN                         ; 3547
                                                54  DD 00114          PUSHL   YES_DEFAULT                              ;
                                                28  BB 00116          PUSHR   #^M<R3,R5>                       ; 3546
                                      7E        0C  AC  7D 00118          MOVQ    FLAGS, -(SP)                     ; 3545
                                                08  AC  DD 0011C          PUSHL   STRING_DEST                      ;
                          00000000G     00        07  FB 0011F          CALLS   #7, COB$$ACC_CONVERT                     ;
                                      56        50  D0 00126          MOVL    R0, CONV_OK                              ;
                                                30  11 00129          BRB     19$                                      ;
    55      08  BC              10              00  ED 0012B 14$:  CMPZV   #0, #16, @STRING_DEST, CHARS_READ ; 3558
                                                05  15 00131          BLEQ    15$                                      ;
                                      6E        55  D0 00133          MOVL    CHARS_READ, COPY_NUM             ; 3560
                                                04  11 00136          BRB     16$                                      ;
                                      6E        08  BC  3C 00138 15$:  MOVZWL  @STRING_DEST, COPY_NUM           ; 3562
                                      09        54  E9 0013C 16$:  BLBC    YES_DEFAULT, 17$                 ; 3565
                                      50        10  AC  D0 0013F          MOVL    DEFAULT, R0                      ; 3566
                                                04  A0  DD 00143          PUSHL   4(R0)                            ;
                                                03  11 00146          BRB     18$                                      ;
                                                04  A3  DD 00148 17$:  PUSHL   4(R3)                            ; 3567
                                                04  AE  9F 0014B 18$:  PUSHAB  COPY_NUM                         ; 3564
                                                08  AC  DD 0014E          PUSHL   STRING_DEST                      ;
                          00000000G     00        03  FB 00151          CALLS   #3, STR$COPY_R                           ;
                                      56        01  D0 00158          MOVL    #1, CONV_OK                              ; 3569
                                                22  12 0015B 19$:  BNEQ    21$                                      ; 3576
                12      0C   AC                  08  E1 0015D          BBC     #11, FLAGS, 20$                  ; 3591
                                            0C  AC  DD 00162          PUSHL   FLAGS                            ; 3598
                                                02  DD 00165          PUSHL   #2                                       ;
                                      6A        02  FB 00167          CALLS   #2, COB$$RMS_PUT_BYTE                    ;
                                            0C  AC  DD 0016A          PUSHL   FLAGS                            ; 3599
                          0000V         CF        01  FB 0016D          CALLS   #1, COB$$RPG_CLEAN_UP                    ;
                                                2A  11 00172          BRB     26$                                      ; 3600
                                      02        05  AC  91 00174 20$:  CMPB    UNIT+1, #2                       ; 3603
                                                24  13 00178          BEQL    26$                                      ;
                                                59  DD 0017A          PUSHL   R9                               ; 3617
                                      67        01  FB 0017C          CALLS   #1, LIB$STOP                             ;
                                            14  AC  D5 0017F 21$:  TSTL    LENGTH                                   ; 3625
```

```
                                    04  13  00182                 BEQL    22$                                                     ;
                              14    BC  55  D0  00184                 MOVL    CHARS_READ, @LENGTH                             ;3627
                        05    0C    AC  0A  E1  00188 22$:            BBC     #10, FLAGS, 24$                                 ;3642
                              68      05  90  0018D 23$:              MOVB    #5, COB$$AB_PREV                                ;3644
                                      08  11  00190                 BRB     25$                                             ;
                                  0C  AC  DD  00192 24$:              PUSHL   FLAGS                                           ;3646
                                      7E  D4  00195                 CLRL    -(SP)                                           ;
                                  6A      02  FB  00197             CALLS   #2, COB$$RMS_PUT_BYTE                           ;
                                  50      01  D0  0019A 25$:         MOVL    #1, R0                                          ;3648
                                          04  0019D                 RET                                                     ;
                                      50  D4  0019E 26$:            CLRL    R0                                              ;3649
                                          04  001A0                 RET                                                     ;

; Routine Size:  417 bytes,    Routine Base:  _COB$CODE + 09AA
```

```
;  2139          3650  1 %SBTTL 'COB$$OPEN_IN - Open for INPUT'
;  2140          3651  1 GLOBAL ROUTINE COB$$OPEN_IN (UNIT, RPG): NOVALUE =
;  2141          3652  1
;  2142          3653  1 !++
;  2143          3654  1 ! FUNCTIONAL DESCRIPTION:
;  2144          3655  1 !
;  2145          3656  1 !       Open a file for reading, given its unit number.
;  2146          3657  1 !
;  2147          3658  1 ! FORMAL PARAMETERS:
;  2148          3659  1 !
;  2149          3660  1 !       UNIT.rl.v       integer unit number designating the device
;  2150          3661  1 !                       from which the string is to be read.
;  2151          3662  1 !
;  2152          3663  1 !       RPG.rl.v        = 1 if RPG is calling this routine
;  2153          3664  1 !                       = 0 if COBOL is calling this routine
;  2154          3665  1 !
;  2155          3666  1 ! IMPLICIT INPUTS:
;  2156          3667  1 !
;  2157          3668  1 !       NONE
;  2158          3669  1 !
;  2159          3670  1 ! IMPLICIT OUTPUTS:
;  2160          3671  1 !
;  2161          3672  1 !       NONE
;  2162          3673  1 !
;  2163          3674  1 ! ROUTINE VALUE:
;  2164          3675  1 !
;  2165          3676  1 !       NONE
;  2166          3677  1 !
;  2167          3678  1 ! SIDE EFFECTS:
;  2168          3679  1 !
;  2169          3680  1 !       NONE
;  2170          3681  1 !
;  2171          3682  1 !--
;  2172          3683  1
;  2173          3684  2     BEGIN
;  2174          3685  2     LITERAL
;  2175          3686  2         MAX_BUF =       MAX(LNMSC_NAMLENGTH, NAMSC_MAXRSS);
;  2176          3687  2     LOCAL
;  2177          3688  2         FAB:            $FAB_DECL,
;  2178          3689  2         NAM:            $NAM_DECL,
;  2179          3690  2         RAB:            REF $RAB_DECL,
;  2180          3691  2         FILE_NAME:      BLOCK[8, BYTE],         ! Descriptor for the file name
;  2181          3692  2         TRANSLATE:      BLOCK[8, BYTE],
;  2182          3693  2         P:              REF VECTOR[,BYTE],
;  2183          3694  2         RSLBUF:         VECTOR[MAX_BUF,BYTE],
;  2184          3695  2         STATUS;
;  2185          3696  2
;  2186          3697  2
;  2187          3698  2     ! Determine whether the COB$xxx name is defined.
;  2188          3699  2     ! If so, use it.  If not, use the corresponding SYS$xxx name.
;  2189          3700  2     !
;  2190          3701  2     TRANSLATE[DSC$B_DTYPE]   = DSC$K_DTYPE_T;
;  2191          3702  2     TRANSLATE[DSC$B_CLASS]   = DSC$K_CLASS_S;
;  2192          3703  2     TRANSLATE[DSC$W_LENGTH]  = MAX_BUF;
;  2193          3704  2     TRANSLATE[DSC$A_POINTER] = RSLBUF;
;  2194          3705  2
;  2195          3706  2     !+
```

```
;  2196          3707  2     !  If VAX RPG is calling this routine, bypass COB_TABLE.
;  2197          3708  2     !-
;  2198          3709  2
;  2199          3710  2     IF .RPG
;  2200          3711  2     THEN
;  2201          3712  2         BEGIN                                       ! Use the SYS$xxx logical
;  2202          3713  2         P = .SYS_TABLE[.UNIT] + BASE;
;  2203          3714  2         FILE_NAME[DSC$W_LENGTH] = .P[0];
;  2204          3715  2         FILE_NAME[DSC$A_POINTER] = P[1];
;  2205          3716  2         END
;  2206          3717  2     ELSE
;  2207          3718  2         BEGIN                                       ! Use the COB$xxx logical
;  2208          3719  2         P = .COB_TABLE[.UNIT] + BASE;
;  2209          3720  2         FILE_NAME[DSC$B_DTYPE]   = DSC$K_DTYPE_T;
;  2210          3721  2         FILE_NAME[DSC$B_CLASS]   = DSC$K_CLASS_S;
;  2211          3722  2         FILE_NAME[DSC$W_LENGTH]  = .P[0];
;  2212          3723  2         FILE_NAME[DSC$A_POINTER] = P[1];
;  2213          3724  2         IF $TRNLOG(LOGNAM = FILE_NAME, RSLBUF = TRANSLATE) NEQ SS$_NORMAL
;  2214          3725  3         THEN
;  2215          3726  4             BEGIN                                   ! Use the SYS$xxx logical
;  2216          3727  4             P = .SYS_TABLE[.UNIT] + BASE;
;  2217          3728  4             FILE_NAME[DSC$W_LENGTH] = .P[0];
;  2218          3729  4             FILE_NAME[DSC$A_POINTER] = P[1];
;  2219          3730  3             END;
;  2220          3731  2         END ;
;  2221          3732  2
;  2222        P 3733  2     $FAB_INIT(
;  2223        P 3734  2         FAB = FAB,
;  2224        P 3735  2         NAM = NAM,
;  2225        P 3736  2         FAC = <GET,PUT>,
;  2226        P 3737  2         FNA = .FILE_NAME[DSC$A_POINTER],
;  2227        P 3738  2         FNS = .FILE_NAME[DSC$W_LENGTH],
;  2228          3739  2         FOP = SQO);
;  2229          3740  2
;  2230        P 3741  2     $NAM_INIT(
;  2231        P 3742  2         NAM = NAM,
;  2232        P 3743  2         ESA = RSLBUF,
;  2233        P 3744  2         ESS = NAM$C_MAXRSS,
;  2234        P 3745  2         RSA = RSLBUF,
;  2235          3746  2         RSS = NAM$C_MAXRSS);
;  2236          3747  2
;  2237          3748  2     STATUS = $OPEN(FAB = FAB);
;  2238          3749  2     IF (TRANSLATE[DSC$W_LENGTH] = .NAM[NAM$B_RSL]) EQL 0 THEN
;  2239          3750  2     IF (TRANSLATE[DSC$W_LENGTH] = .NAM[NAM$B_ESL]) EQL 0
;  2240          3751  2     THEN
;  2241          3752  2         BEGIN
;  2242          3753  2         TRANSLATE[DSC$W_LENGTH] = .FAB[FAB$B_FNS];
;  2243          3754  2         TRANSLATE[DSC$A_POINTER]= .FAB[FAB$L_FNA];
;  2244          3755  2         END;
;  2245          3756  2
;  2246          3757  2
;  2247          3758  2     IF NOT .STATUS
;  2248          3759  2     THEN
;  2249          3760  2         LIB$STOP(COB$_ERRDURACC, 1, TRANSLATE, .FAB[FAB$L_STS], .FAB[FAB$L_STV]);
;  2250          3761  2
;  2251          3762  2
;  2252          3763  3     IF NOT (STATUS = LIB$GET_VM(%REF(RAB$C_BLN + 8 + .NAM[NAM$B_RSL]), RAB))
```

```
;  2253          3764  2      THEN
;  2254          3765  2          LIB$STOP(COB$_FAIGET_VM, 0, .STATUS);
;  2255          3766  2
;  2256          3767  2
;  2257          3768  2      ! Save a descriptor for the resultant file name string,
;  2258          3769  2      ! and the string itself, after the RAB
;  2259          3770  2      !
;  2260          3771  2      BEGIN
;  2261          3772  2      LOCAL
;  2262          3773  2          Q: REF BLOCK[,BYTE];
;  2263          3774  2      Q = .RAB + RAB$C_BLN;
;  2264          3775  2      Q[DSC$B_DTYPE]   = DSC$K_DTYPE_T;
;  2265          3776  2      Q[DSC$B_CLASS]   = DSC$K_CLASS_S;
;  2266          3777  2      Q[DSC$W_LENGTH]  = .TRANSLATE[DSC$W_LENGTH];
;  2267          3778  2      Q[DSC$A_POINTER] = .RAB+RAB$C_BLN+8;
;  2268          3779  2      CH$MOVE( .Q[DSC$W_LENGTH], .TRANSLATE[DSC$A_POINTER], .RAB+RAB$C_BLN+8 );
;  2269          3780  2      END;
;  2270          3781  2
;  2271          3782  2      !+
;  2272          3783  2      ! Initiate terminal XABTRM and include it in the RAB.
;  2273          3784  2      !-
;  2274          3785  2
;  2275        P 3786  2      $XABTRM_INIT ( XAB         = XABTRM,
;  2276        P 3787  2                     ITMLST      = XAB_ITMLST,
;  2277          3788  2                     ITMLST_LEN = %ALLOCATION (XAB_ITMLST)-4 ) ;  ! $ITMLST_DECL
;  2278          3789  2                                                                  ! adds extra 4
;  2279        P 3790  2      $RAB_INIT(
;  2280        P 3791  2          RAB = .RAB,
;  2281        P 3792  2          FAB = FAB,
;  2282          3793  2          XAB = XABTRM);
;  2283          3794  2
;  2284          3795  2      IF NOT $CONNECT(RAB = .RAB)
;  2285          3796  2      THEN
;  2286          3797  2          LIB$STOP(COB$_ERRDURACC, 1, .RAB+RAB$C_BLN, .RAB[RAB$L_STS], .RAB[RAB$L_STV]);
;  2287          3798  2
;  2288          3799  2      COB$$AL_WRITE_RAB[.UNIT] = .RAB;
;  2289          3800  2      COB$$AW_WRITE_IFI[.UNIT] = .FAB[FAB$W_IFI];
;  2290          3801  2
;  2291          3802  1      END;                                        !  End of COB$$OPEN_IN


                                                    $RMS_PTR=           XABTRM
                                                                .EXTRN  SYS$TRNLOG, SYS$OPEN
                                                                .EXTRN  SYS$CONNECT

                                                OFFC 00000          .ENTRY  COB$$OPEN_IN, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 3651
                                                                            R10,R11
                                                                                                                       ;
                    5B 00000000G   00    9E 00002               MOVAB   LIB$STOP, R11                                 ;
                    5A 00000000'   EF    9E 00009               MOVAB   $RMS_PTR, R10                                 ;
                    5E      FE38   CE    9E 00010               MOVAB   -456(SP), SP                                  ;
            FF40    CD 010E00FF    8F    D0 00015               MOVL    #17694975, TRANSLATE                          ; 3703
            FF44    CD        08   AE    9E 0001E               MOVAB   RSLBUF, TRANSLATE+4                           ; 3704
                    58        04   AC    D0 00024               MOVL    UNIT, R8                                      ; 3713
        57          58             02    78 00028               ASHL    #2, R8, R7                                    ;
                    3A        08   AC    E8 0002C               BLBS    RPG, 1$                                       ; 3710
                    50      F481   CF    9E 00030               MOVAB   BASE, R0                                      ; 3719
```

```
                                F4E3 CF47  9F 00035          PUSHAB  COB_TABLE[R7]                                   ;
                          52          50    9E C1 0003A          ADDL3   @(SP)+, R0, P                                   ;
                              FF4A    CD  010E  8F B0 0003E          MOVW    #270, FILE_NAME+2                       ; 3720
                              FF48    CD    62  9B 00045          MOVZBW  (P), FILE_NAME                          ; 3722
                              FF4C    CD    01  A2 9E 0004A          MOVAB   1(R2), FILE_NAME+4                      ; 3723
                                            7E  7C 00050          CLRQ    -(SP)                                   ; 3724
                                            7E  D4 00052          CLRL    -(SP)                                   ;
                                F40     CD  9F 00054          PUSHAB  TRANSLATE                               ;
                                            7E  D4 00058          CLRL    -(SP)                                   ;
                                F48     CD  9F 0005A          PUSHAB  FILE_NAME                               ;
                    00000000G   00          06  FB 0005E          CALLS   #6, SYS$TRNLOG                          ;
                                01          50  D1 00065          CMPL    R0, #1                                  ;
                                            19  13 00068          BEQL    2$                                      ;
                                50      F447    CF 9E 0006A 1$:       MOVAB   BASE, R0                                ; 3727
                                F511 CF47  9F 0006F          PUSHAB  SYS_TABLE[R7]                           ;
                          52          50    9E C1 00074          ADDL3   @(SP)+, R0, P                                   ;
                              FF48    CD    62  9B 00078          MOVZBW  (P), FILE_NAME                          ; 3728
                              FF4C    CD    01  A2 9E 0007D          MOVAB   1(R2), FILE_NAME+4                      ; 3729
    0050    8F        00              6E    00  2C 00083 2$:       MOVC5   #0, (SP), #0, #80, $RMS_PTR             ; 3739
                                        B0    AD 0008A
                                  B0    AD  5003  8F B0 0008C          MOVW    #20483, $RMS_PTR                        ;
                                  B4    AD    40  8F 9A 00092          MOVZBL  #64, $RMS_PTR+4                         ;
                                  C6    AD        03 90 00097          MOVB    #3, $RMS_PTR+22                         ;
                                  CF    AD        02 90 0009B          MOVB    #2, $RMS_PTR+31                         ;
                                  D8    AD  FF50  CD 9E 0009F          MOVAB   NAM, $RMS_PTR+40                        ;
                                  DC    AD  FF4C  CD D0 000A5          MOVL    FILE_NAME+4, $RMS_PTR+44                ;
                                  E4    AD  FF48  CD 90 000AB          MOVB    FILE_NAME, $RMS_PTR+52                  ;
    0060    8F        00              6E        00 2C 000B1          MOVC5   #0, (SP), #0, #96, $RMS_PTR             ; 3746
                                            FF50  CD 000B8
                                FF50    CD  6002  8F B0 000BB          MOVW    #24578, $RMS_PTR                        ;
                                FF52    CD        01 8E 000C2          MNEGB   #1, $RMS_PTR+2                          ;
                                FF54    CD    08  AE 9E 000C7          MOVAB   RSLBUF, $RMS_PTR+4                      ;
                                FF5A    CD        01 8E 000CD          MNEGB   #1, $RMS_PTR+10                         ;
                                FF5C    CD    08  AE 9E 000D2          MOVAB   RSLBUF, $RMS_PTR+12                     ;
                                            B0    AD 9F 000D8          PUSHAB  FAB                                     ; 3748
                    00000000G   00              01 FB 000DB          CALLS   #1, SYS$OPEN                            ;
                                52              50 D0 000E2          MOVL    R0, STATUS                              ;
                          FF40  CD        FF53  CD 9B 000E5          MOVZBW  NAM+3, TRANSLATE                        ; 3749
                                            15  12 000EC          BNEQ    3$                                      ;
                          FF40  CD        FF5B  CD 9B 000EE          MOVZBW  NAM+11, TRANSLATE                       ; 3750
                                            0C  12 000F5          BNEQ    3$                                      ;
                          FF40  CD          E4  AD 9B 000F7          MOVZBW  FAB+52, TRANSLATE                       ; 3753
                          FF44  CD          DC  AD D0 000FD          MOVL    FAB+44, TRANSLATE+4                     ; 3754
                                13              52  E8 00103 3$:       BLBS    STATUS, 4$                              ; 3758
                                7E          B8  AD 7D 00106          MOVQ    FAB+8, -(SP)                            ; 3760
                                        FF40    CD 9F 0010A          PUSHAB  TRANSLATE                               ;
                                                01 DD 0010E          PUSHL   #1                                      ;
                                    00000000G   8F DD 00110          PUSHL   #COB$_ERRDURACC                         ;
                                6B              05 FB 00116          CALLS   #5, LIB$STOP                            ;
                                            04  AE 9F 00119 4$:       PUSHAB  RAB                                     ; 3763
                                  04    AE  FF53  CD 9A 0011C          MOVZBL  NAM+3, 4(SP)                            ;
                                  04    AE 0000004C 8F C0 00122          ADDL2   #76, 4(SP)                              ;
                                            04  AE 9F 0012A          PUSHAB  4(SP)                                   ;
                    00000000G   00              02 FB 0012D          CALLS   #2, LIB$GET_VM                          ;
                                52              50 D0 00134          MOVL    R0, STATUS                              ;
                                0D              52 E8 00137          BLBS    STATUS, 5$                              ;
                                                52 DD 0013A          PUSHL   STATUS                                  ; 3765
```

```
                                                    7E  D4 0013C          CLRL    -(SP)                                          ;
                                        00000000G   8F  DD 0013E          PUSHL   #COB$_FAIGET_VM                                ;
                                              6B        03  FB 00144          CALLS   #3, LIB$STOP                           ;
                                              56    04  AE  D0 00147 5$:     MOVL    RAB, R6                                ; 3774
                                              59    44  A6  9E 0014B         MOVAB   68(R6), R9                             ;
                                              50        59  D0 0014F         MOVL    R9, Q                                  ;
                                        02    A0  010E  8F  B0 00152         MOVW    #270, 2(Q)                             ; 3775
                                              60  FF40  CD  B0 00158         MOVW    TRANSLATE, (Q)                         ; 3777
                                        04    A0    4C  A6  9E 0015D         MOVAB   76(R6), 4(Q)                           ; 3778
                                  4C    A6  FF44  DD      60  28 00162         MOVC3   (Q), @TRANSLATE+4, 76(R6)              ; 3779
                              24        00        6E      00  2C 00169         MOVC5   #0, (SP), #0, #36, $RMS_PTR            ; 3788
                                                        6A      0016E                                                          ;
                                              6A  241F  8F  B0 0016F         MOVW    #9247, $RMS_PTR                        ;
                                        08    AA    24  AA  9E 00174         MOVAB   XAB_ITMLST, $RMS_PTR+8                 ;
                                        0C    AA        18  B0 00179         MOVW    #24, $RMS_PTR+12                       ;
                        0044    8F        00        6E      00  2C 0017D         MOVC5   #0, (SP), #0, #68, (R6)                ; 3793
                                                        66      00184                                                          ;
                                              66  4401  8F  B0 00185         MOVW    #17409, (R6)                           ;
                                        3C    A6    B0  AD  9E 0018A         MOVAB   FAB, 60(R6)                            ;
                                        40    A6        6A  9E 0018F         MOVAB   XABTRM, 64(R6)                         ;
                                                        56  DD 00193         PUSHL   R6                                     ; 3795
                                  00000000G   00        01  FB 00195         CALLS   #1, SYS$CONNECT                        ;
                                                        11  50  E8 0019C         BLBS    R0, 6$                                 ;
                                              7E    08  A6  7D 0019F         MOVQ    8(R6), -(SP)                           ; 3797
                                                        59  DD 001A3         PUSHL   R9                                     ;
                                                        01  DD 001A5         PUSHL   #1                                     ;
                                        00000000G       8F  DD 001A7         PUSHL   #COB$_ERRDURACC                        ;
                                              6B        05  FB 001AD         CALLS   #5, LIB$STOP                           ;
                                  00000000G0047         9F  001B0 6$:        PUSHAB  COB$$AL_WRITE_RAB[R7]                  ; 3799
                                              9E        56  D0 001B7         MOVL    R6, @(SP)+                             ;
                            00000000G0048       B2    AD  B0 001BA         MOVW    FAB+2, COB$$AW_WRITE_IFI[R8]           ; 3800
                                                        04  001C3         RET                                            ; 3802

; Routine Size:  452 bytes,    Routine Base:  _COB$CODE + 0B4B
```

```
;  2293          3803  1 %SBTTL 'COB$$RMS_GET - Perform an RMS $GET Service'
;  2294          3804  1 ROUTINE COB$$RMS_GET ( RAB      :  REF $RAB_DECL,
;  2295          3805  1                        FUNC_VAL,
;  2296          3806  1                        LENGTH,
;  2297          3807  1                        BUFFER
;  2298          3808  1                      )  :  NOVALUE  =
;  2299          3809  1
;  2300          3810  1 !++
;  2301          3811  1 ! FUNCTIONAL DESCRIPTION:
;  2302          3812  1 !
;  2303          3813  1 !
;  2304          3814  1 ! FORMAL PARAMETERS:
;  2305          3815  1 !
;  2306          3816  1 !
;  2307          3817  1 ! IMPLICIT INPUTS:
;  2308          3818  1 !
;  2309          3819  1 !       NONE
;  2310          3820  1 !
;  2311          3821  1 ! IMPLICIT OUTPUTS:
;  2312          3822  1 !
;  2313          3823  1 !       NONE
;  2314          3824  1 !
;  2315          3825  1 ! ROUTINE VALUE:
;  2316          3826  1 ! COMPLETION CODES:
;  2317          3827  1 !
;  2318          3828  1 !       NONE
;  2319          3829  1 !
;  2320          3830  1 ! SIDE EFFECTS:
;  2321          3831  1 !
;  2322          3832  1 !--
;  2323          3833  1
;  2324          3834  2     BEGIN
;  2325          3835  2
;  2326        P 3836  2     $ITMLST_INIT (ITMLST = XAB_ITMLST,                      ! Item list for $GET
;  2327        P 3837  2                   (ITMCOD = TRM$_MODIFIERS,
;  2328        P 3838  2                    BUFSIZ = 0,
;  2329        P 3839  2                    BUFADR = .FUNC_VAL),
;  2330        P 3840  2                   (ITMCOD = TRM$_TERM,
;  2331        P 3841  2                    BUFSIZ = 20,
;  2332          3842  2                    BUFADR = MASK_VECTOR) ) ;
;  2333          3843  2
;  2334          3844  2     RAB [RAB$W_USZ] = .LENGTH ;
;  2335          3845  2     RAB [RAB$L_UBF] = .BUFFER ;
;  2336          3846  2     RAB [RAB$V_ETO] = 1 ;                                   ! Extended Terminal $GET
;  2337          3847  2     RAB [RAB$L_XAB] = XABTRM ;
;  2338          3848  2     WHILE $GET (RAB = .RAB) EQL RMS$_RSA DO $WAIT (RAB = .RAB) ;
;  2339          3849  2
;  2340          3850  2     IF NOT .RAB [RAB$L_STS]
;  2341          3851  2     THEN
;  2342          3852  2         !+
;  2343          3853  2         ! These are special case status that will be handled later.
;  2344          3854  2         !-
;  2345          3855  3         IF (.RAB [RAB$L_STS] NEQ RMS$_BES AND               ! Bad Escape Sequence
;  2346          3856  3             .RAB [RAB$L_STS] NEQ RMS$_EOF AND               ! End Of File
;  2347          3857  3             .RAB [RAB$L_STS] NEQ RMS$_PES AND               ! Partial Escape Seq
;  2348          3858  3             .RAB [RAB$L_STS] NEQ RMS$_RTB AND               ! Record Too Big
;  2349          3859  3             .RAB [RAB$L_STS] NEQ RMS$_TNS )                 ! Terminator Not Seen
```

```
;  2350          3860  2          THEN
;  2351          3861  2              LIB$STOP (COB$_ERRDURACC, 1, .RAB + RAB$C_BLN, .RAB [RAB$L_STS],
;  2352          3862  2                                                             .RAB [RAB$L_STV] ) ;
;  2353          3863  1      END ;



                                        000C 00000 COB$$RMS_GET:
                                                    .WORD   Save R2,R3                                      ; 3804
                      53 00000000'  EF  9E 00002            MOVAB   XAB_ITMLST, R3
                      50            63  9E 00009            MOVAB   XAB_ITMLST, $$ITMBLKPTR                 ; 3842
                                    80  D4 0000C            CLRL    ($$ITMBLKPTR)+
                      80       08   AC  D0 0000E            MOVL    FUNC_VAL, ($$ITMBLKPTR)+
                                    80  D4 00012            CLRL    ($$ITMBLKPTR)+
                      80 00030014   8F  D0 00014            MOVL    #196628, ($$ITMBLKPTR)+
                      80       1C   A3  9E 0001B            MOVAB   MASK_VECTOR, ($$ITMBLKPTR)+
                                    80  7C 0001F            CLRQ    ($$ITMBLKPTR)+
                      52       04   AC  D0 00021            MOVL    RAB, R2                                ; 3844
                20    A2       0C   AC  B0 00025            MOVW    LENGTH, 32(R2)
                24    A2       10   AC  D0 0002A            MOVL    BUFFER, 36(R2)                         ; 3845
                07    A2            10  88 0002F            BISB2   #16, 7(R2)                             ; 3846
                40    A2       DC   A3  9E 00033            MOVAB   XABTRM, 64(R2)                         ; 3847
                                    52  DD 00038 1$:         PUSHL   R2                                     ; 3848
          00000000G   00            01  FB 0003A            CALLS   #1, SYS$GET
          000182DA    8F            50  D1 00041            CMPL    R0, #99034
                                    0B  12 00048            BNEQ    2$
                                    52  DD 0004A            PUSHL   R2
          00000000G   00            01  FB 0004C            CALLS   #1, SYS$WAIT
                                    E3  11 00053            BRB     1$
                      50       08   A2  D0 00055 2$:         MOVL    8(R2), R0                              ; 3850
                      44            50  E8 00059            BLBS    R0, 3$
          000181C0    8F            50  D1 0005C            CMPL    R0, #98752                             ; 3855
                                    3B  13 00063            BEQL    3$
          0001827A    8F            50  D1 00065            CMPL    R0, #98938                             ; 3856
                                    32  13 0006C            BEQL    3$
          000181C8    8F            50  D1 0006E            CMPL    R0, #98760                             ; 3857
                                    29  13 00075            BEQL    3$
          000181A8    8F            50  D1 00077            CMPL    R0, #98728                             ; 3858
                                    20  13 0007E            BEQL    3$
          000181B8    8F            50  D1 00080            CMPL    R0, #98744                             ; 3859
                                    17  13 00087            BEQL    3$
                                0C  A2  DD 00089            PUSHL   12(R2)                                 ; 3862
                                    50  DD 0008C            PUSHL   R0                                     ; 3861
                                44  A2  9F 0008E            PUSHAB  68(R2)
                                    01  DD 00091            PUSHL   #1
                            00000000G   8F  DD 00093        PUSHL   #COB$_ERRDURACC
          00000000G   00            05  FB 00099            CALLS   #5, LIB$STOP
                                        04 000A0 3$:        RET                                            ; 3863

; Routine Size:  161 bytes,    Routine Base:  _COB$CODE + 0D0F
```

```
;  2355          3864  1 %SBTTL 'COB$$RMS_PUT_BYTE - Perform an RMS $PUT Service'
;  2356          3865  1 ROUTINE COB$$RMS_PUT_BYTE ( WHICH, FLAGS )  :  NOVALUE  =
;  2357          3866  1
;  2358          3867  1 !++
;  2359          3868  1 ! FUNCTIONAL DESCRIPTION:
;  2360          3869  1 !
;  2361          3870  1 !       This routine writes a one byte buffer to the terminal. Either a
;  2362          3871  1 !       Carriage Return, Linefeed or Ring the Terminal Bell, depending
;  2363          3872  1 !       on the value of WHICH.
;  2364          3873  1 !
;  2365          3874  1 ! FORMAL PARAMETERS:
;  2366          3875  1 !
;  2367          3876  1 !       WHICH.rl.v      if 0, write Linefeed to terminal
;  2368          3877  1 !                       if 1, write Carriage Return to terminal
;  2369          3878  1 !                       if 2, ring terminal bell
;  2370          3879  1 !
;  2371          3880  1 !       FLAGS.rlu.v     Screen enhancement flag
;  2372          3881  1 !
;  2373          3882  1 ! IMPLICIT INPUTS:
;  2374          3883  1 !
;  2375          3884  1 !       NONE
;  2376          3885  1 !
;  2377          3886  1 ! IMPLICIT OUTPUTS:
;  2378          3887  1 !
;  2379          3888  1 !       NONE
;  2380          3889  1 !
;  2381          3890  1 ! ROUTINE VALUE:
;  2382          3891  1 ! COMPLETION CODES:
;  2383          3892  1 !
;  2384          3893  1 !       NONE
;  2385          3894  1 !
;  2386          3895  1 ! SIDE EFFECTS:
;  2387          3896  1 !
;  2388          3897  1 !       NONE
;  2389          3898  1 !--
;  2390          3899  1
;  2391          3900  2     BEGIN
;  2392          3901  2     LOCAL
;  2393          3902  2         RAB     :  REF $RAB_DECL,
;  2394          3903  2         CR_BUF  :  VECTOR [1,BYTE] ;
;  2395          3904  2
;  2396          3905  2     IF .COB$ACC_TERM_TYPE NEQ UNKNOWN
;  2397          3906  2     THEN
;  2398          3907  2         BEGIN
;  2399          3908  2
;  2400          3909  2         SELECTONE .WHICH OF
;  2401          3910  2             SET
;  2402          3911  2                 [0] :   CR_BUF [0] = CR ;               ! Carriage Return
;  2403          3912  2
;  2404          3913  2                 [1] :   CR_BUF [0] = LF ;               ! Linefeed
;  2405          3914  2
;  2406          3915  2                 [2] :   CR_BUF [0] = BELL ;             ! Bell
;  2407          3916  2             TES ;
;  2408          3917  2
;  2409          3918  2         COB$$AB_USPCODE [0] = 0 ;
;  2410          3919  2         COB$$AB_USPCODE [1] = 0 ;
;  2411          3920  3
```

```
;  2412          3921  3          IF .COB$$AL_WRITE_RAB [1] EQL 0
;  2413          3922  3          THEN
;  2414          3923  4              BEGIN
;  2415          3924  4              !+
;  2416          3925  4              !  Open SYS$OUTPUT.  Second parameter tells COB$$OPEN_OUT whether
;  2417          3926  4              !  VAX COBOL (0) or VAX RPG (1) is the caller.
;  2418          3927  4              !  VMS V4 defines SYS$INPUT as read only, therefore any $PUTs must
;  2419          3928  4              !  be made through SYS$OUTPUT.  When a terminal is the input device
;  2420          3929  4              !  for an ACCEPT it is also the OUTPUT device, and must be OPENed
;  2421          3930  4              !  for both.
;  2422          3931  4              !-
;  2423          3932  4              COB$$OPEN_OUT ( 1,
;  2424          3933  4                              IF ( .FLAGS AND V_COB_RPG ) NEQ 0
;  2425          3934  4                              THEN 1
;  2426          3935  4                              ELSE 0 ) ;
;  2427          3936  4              END ;
;  2428          3937          RAB = .COB$$AL_WRITE_RAB [1] ;
;  2429          3938          RAB [RAB$L_RBF] = CR_BUF [0] ;
;  2430          3939          RAB [RAB$W_RSZ] = 1 ;
;  2431          3940          WHILE $PUT (RAB = .RAB) EQL RMS$_RSA DO $WAIT (RAB = .RAB) ;
;  2432          3941
;  2433          3942          IF NOT .RAB [RAB$L_STS]
;  2434          3943          THEN
;  2435          3944              LIB$STOP ( COB$_ERRDURACC, 1, .RAB + RAB$C_BLN,
;  2436          3945                         .RAB [RAB$L_STS], .RAB [RAB$L_STV] ) ;
;  2437          3946  2       END ;
;  2438          3947  1   END ;                                        ! End of COB$$RMS_PUT_BYTE


                                        .EXTRN  SYS$PUT

                                000C 00000 COB$$RMS_PUT_BYTE:
                                           .WORD   Save R2,R3                                   ;  3865
                    53 00000000G   00  9E 00002            MOVAB   COB$$AL_WRITE_RAB+4, R3      ;
                    5E             04  C2 00009            SUBL2   #4, SP                       ;
                       00000000G   00  D5 0000C            TSTL    COB$ACC_TERM_TYPE            ;  3905
                                   7D  13 00012            BEQL    9$                           ;
                    50         04  AC  D0 00014            MOVL    WHICH, R0                    ;  3909
                                   05  12 00018            BNEQ    1$                           ;  3911
                    6E                 0D  90 0001A            MOVB    #13, CR_BUF                  ;
                                       12  11 0001D            BRB     3$                           ;
                    01                 50  D1 0001F 1$:        CMPL    R0, #1                       ;  3913
                                       05  12 00022            BNEQ    2$                           ;
                    6E                 0A  90 00024            MOVB    #10, CR_BUF                  ;
                                       08  11 00027            BRB     3$                           ;
                    02                 50  D1 00029 2$:        CMPL    R0, #2                       ;  3915
                                       03  12 0002C            BNEQ    3$                           ;
                    6E                 07  90 0002E            MOVB    #7, CR_BUF                   ;
                       00000000G       00  B4 00031 3$:        CLRW    COB$$AB_USPCODE              ;  3918
                                       63  D5 00037            TSTL    COB$$AL_WRITE_RAB+4          ;  3921
                                       14  12 00039            BNEQ    6$                           ;
          04       08   AC             0B  E1 0003B            BBC     #11, FLAGS, 4$               ;  3933
                                       01  DD 00040            PUSHL   #1                           ;
                                       02  11 00042            BRB     5$                           ;
                                       7E  D4 00044 4$:        CLRL    -(SP)                        ;
                                       01  DD 00046 5$:        PUSHL   #1                           ;  3932
```

```
                          00000000G  00              02  FB 00048              CALLS   #2, COB$$OPEN_OUT                                ;
                                     52              63  D0 0004F 6$:          MOVL    COB$$AL_WRITE_RAB+4, RAB                        ; 3937
                                 28  A2              6E  9E 00052              MOVAB   CR_BUF, -40(RAB)                                ; 3938
                                 22  A2              01  B0 00056              MOVW    #1, 34(RAB)                                     ; 3939
                                                     52  DD 0005A 7$:          PUSHL   RAB                                             ; 3940
                          00000000G  00              01  FB 0005C              CALLS   #1, SYS$PUT                                     ;
                          000182DA   8F              50  D1 00063              CMPL    R0, #99034                                      ;
                                                     0B  12 0006A              BNEQ    8$                                              ;
                                                     52  DD 0006C              PUSHL   RAB                                             ;
                          00000000G  00              01  FB 0006E              CALLS   #1, SYS$WAIT                                    ;
                                                     E3  11 00075              BRB     7$                                              ;
                                     16          08  A2  E8 00077 8$:          BLBS    8(RAB), 9$                                      ; 3942
                                     7E          08  A2  7D 0007B              MOVQ    8(RAB), -(SP)                                   ; 3945
                                                 44  A2  9F 0007F              PUSHAB  68(RAB)                                         ; 3944
                                                     01  DD 00082              PUSHL   #1                                              ;
                                           00000000G  8F  DD 00084              PUSHL   #COB$_ERRDURACC                                 ;
                          00000000G  00              05  FB 0008A              CALLS   #5, LIB$STOP                                    ;
                                                     04  00091 9$:            RET                                                     ; 3947

; Routine Size:  146 bytes,    Routine Base:  _COB$CODE + 0DB0
```

```
;  2440          3948  1 %SBTTL 'COB$$RMS_PUT_BUFFER - Perform RMS $PUT Service'
;  2441          3949  1 ROUTINE COB$$RMS_PUT_BUFFER ( BUFFER,
;  2442          3950  1                               LENGTH,
;  2443          3951  1                               FLAGS )  :  NOVALUE  =
;  2444          3952  1
;  2445          3953  1 !++
;  2446          3954  1 ! FUNCTIONAL DESCRIPTION:
;  2447          3955  1 !
;  2448          3956  1 !       This routine writes buffer of more than one byte to the terminal.
;  2449          3957  1 !
;  2450          3958  1 ! FORMAL PARAMETERS:
;  2451          3959  1 !
;  2452          3960  1 !       BUFFER.rt.r     Holds sequence to write to screen
;  2453          3961  1 !
;  2454          3962  1 !       LENGHT.rlu.v    Length of BUFFER
;  2455          3963  1 !
;  2456          3964  1 !       FLAGS.rlu.v     Screen enhancement flag
;  2457          3965  1 !
;  2458          3966  1 ! IMPLICIT INPUTS:
;  2459          3967  1 !
;  2460          3968  1 !       NONE
;  2461          3969  1 !
;  2462          3970  1 ! IMPLICIT OUTPUTS:
;  2463          3971  1 !
;  2464          3972  1 !       NONE
;  2465          3973  1 !
;  2466          3974  1 ! ROUTINE VALUE:
;  2467          3975  1 ! COMPLETION CODES:
;  2468          3976  1 !
;  2469          3977  1 !       NONE
;  2470          3978  1 !
;  2471          3979  1 ! SIDE EFFECTS:
;  2472          3980  1 !
;  2473          3981  1 !       NONE
;  2474          3982  1 !--
;  2475          3983  1
;  2476          3984  2     BEGIN
;  2477          3985  2     LOCAL
;  2478          3986  2         RAB     :  REF $RAB_DECL ;
;  2479          3987  2
;  2480          3988  2     IF .COB$ACC_TERM_TYPE NEQ UNKNOWN
;  2481          3989  2     THEN
;  2482          3990  3         BEGIN
;  2483          3991  3
;  2484          3992  3         COB$$AB_USPCODE [0] = 0 ;
;  2485          3993  3         COB$$AB_USPCODE [1] = 0 ;
;  2486          3994  3
;  2487          3995  3         IF .COB$$AL_WRITE_RAB [1] EQL 0
;  2488          3996  3         THEN
;  2489          3997  4             BEGIN
;  2490          3998  4             !+
;  2491          3999  4             ! Open SYS$OUTPUT.  Second parameter tells COB$$OPEN_OUT whether
;  2492          4000  4             ! VAX COBOL (0) or VAX RPG (1) is the caller.
;  2493          4001  4             !-
;  2494          4002  4             COB$$OPEN_OUT ( 1,
;  2495          4003  4                             IF ( .FLAGS AND V_COB_RPG ) NEQ 0
;  2496          4004  4                             THEN 1
```

```
;  2497          4005  4                                ELSE 0 ) ;
;  2498          4006  4                            END ;
;  2499          4007  4                        RAB = .COB$$AL_WRITE_RAB [1] ;
;  2500          4008  4                        RAB [RAB$L_RBF] = .BUFFER ;
;  2501          4009  4                        RAB [RAB$W_RSZ] = .LENGTH ;
;  2502          4010  4                        WHILE $PUT (RAB = .RAB) EQL RMS$_RSA DO $WAIT (RAB = .RAB) ;
;  2503          4011  4
;  2504          4012  4                        IF NOT .RAB [RAB$L_STS]
;  2505          4013  4                        THEN
;  2506          4014  4                            LIB$STOP ( COB$_ERRDURACC, 1, .RAB + RAB$C_BLN,
;  2507          4015  4                                       .RAB [RAB$L_STS], .RAB [RAB$L_STV] ) ;
;  2508          4016  4                        END ;
;  2509          4017  1                END ;                                    ! End of COB$$RMS_PUT_BUFFER


                                                        000C 00000 COB$$RMS_PUT_BUFFER:
                                                                   .WORD   Save R2,R3                                  ;  3949
                                    53 00000000G   00   9E 00002           MOVAB   COB$$AL_WRITE_RAB+4, R3             ;
                                       00000000G   00   D5 00009           TSTL    COB$ACC_TERM_TYPE                   ;  3988
                                                   62   13 0000F           BEQL    6$                                  ;
                                       00000000G   00   B4 00011           CLRW    COB$$AB_USPCODE                     ;  3992
                                                   63   D5 00017           TSTL    COB$$AL_WRITE_RAB+4                 ;  3995
                                                   14   12 00019           BNEQ    3$                                  ;
                          04        0C    AC       0B   E1 0001B           BBC     #11, FLAGS, 1$                      ;  4003
                                                   01   DD 00020           PUSHL   #1                                  ;
                                                   02   11 00022           BRB     2$                                  ;
                                                   7E   D4 00024 1$:       CLRL    -(SP)                               ;
                                                   01   DD 00026 2$:       PUSHL   #1                                  ;  4002
                             00000000G   00        02   FB 00028           CALLS   #2, COB$$OPEN_OUT                   ;
                                         52        63   D0 0002F 3$:       MOVL    COB$$AL_WRITE_RAB+4, RAB            ;  4007
                                   28    A2   04   AC   D0 00032           MOVL    BUFFER, 40(RAB)                     ;  4008
                                   22    A2   08   AC   B0 00037           MOVW    LENGTH, 34(RAB)                     ;  4009
                                                   52   DD 0003C 4$:       PUSHL   RAB                                 ;  4010
                             00000000G   00        01   FB 0003E           CALLS   #1, SYS$PUT                         ;
                             000182DA    8F        50   D1 00045           CMPL    R0, #99034                          ;
                                                   0B   12 0004C           BNEQ    5$                                  ;
                                                   52   DD 0004E           PUSHL   RAB                                 ;
                             00000000G   00        01   FB 00050           CALLS   #1, SYS$WAIT                        ;
                                                   E3   11 00057           BRB     4$                                  ;
                                         16   08   A2   E8 00059 5$:       BLBS    8(RAB), 6$                          ;  4012
                                         7E   08   A2   7D 0005D           MOVQ    8(RAB), -(SP)                       ;  4015
                                              44   A2   9F 00061           PUSHAB  68(RAB)                             ;  4014
                                                   01   DD 00064           PUSHL   #1                                  ;
                                   00000000G       8F   DD 00066           PUSHL   #COB$_ERRDURACC                     ;
                             00000000G   00        05   FB 0006C           CALLS   #5, LIB$STOP                        ;
                                                   04   00073 6$:         RET                                         ;  4017

; Routine Size:  116 bytes,    Routine Base:  _COB$CODE + 0E42
```

```
;  2511          4018  1 %SBTTL 'COBS$CONTROL_Z - Handle ^Z'
;  2512          4019  1 ROUTINE COBS$CONTROL_Z ( UNIT            : VECTOR [2, BYTE],
;  2513          4020  1                          KEY             : REF $STRSDESCRIPTOR
;  2514          4021  1                        )  :  NOVALUE =
;  2515          4022  1 !++
;  2516          4023  1 ! FUNCTIONAL DESCRIPTION:
;  2517          4024  1 !
;  2518          4025  1 !       Read UNIT parameter to determine what to do when a Control Z was typed.
;  2519          4026  1 !
;  2520          4027  1 ! FORMAL PARAMETERS:
;  2521          4028  1 !
;  2522          4029  1 !       UNIT.rbu.va     Array of two unsigned byte integers.
;  2523          4030  1 !                       The first byte is the unit number designating the
;  2524          4031  1 !                       device from which the string is to be read.
;  2525          4032  1 !                       The second byte indicates whether the routine should
;  2526          4033  1 !                       abort or return to the calling program.
;  2527          4034  1 !
;  2528          4035  1 !       KEY.wt.ds       Destination of the receiving field of the control key.
;  2529          4036  1 !
;  2530          4037  1 ! IMPLICIT INPUTS:
;  2531          4038  1 !
;  2532          4039  1 !       NONE
;  2533          4040  1 !
;  2534          4041  1 ! IMPLICIT OUTPUTS:
;  2535          4042  1 !
;  2536          4043  1 !       NONE
;  2537          4044  1 !
;  2538          4045  1 ! ROUTINE VALUE:
;  2539          4046  1 !
;  2540          4047  1 !
;  2541          4048  1 ! SIDE EFFECTS:
;  2542          4049  1 !
;  2543          4050  1 !       NONE
;  2544          4051  1 !--
;  2545          4052  2 BEGIN
;  2546          4053  2     LOCAL
;  2547          4054  2         TERM_PTR,                               ! Points to terminator
;  2548          4055  2         CZ_PTR ;                                ! Needed for CH$MOVE
;  2549          4056  2
;  2550          4057  2 !+
;  2551          4058  2 ! CONTROL Z - read UNIT parameter to determine what to do.
;  2552          4059  2 !
;  2553          4060  2 !       Byte 2 of               Ctrl z
;  2554          4061  2 !         UNIT
;  2555          4062  2 !
;  2556          4063  2 !           0                   abort
;  2557          4064  2 !           1 ( at end )        return
;  2558          4065  2 !           2 ( on exception )  return
;  2559          4066  2 !-
;  2560          4067  2
;  2561          4068  2     IF .UNIT [1]  EQL  0
;  2562          4069  2     THEN
;  2563          4070  2         LIB$STOP ( COBS_EOFON_ACC )             ! Abort
;  2564          4071  2     ELSE
;  2565          4072  3         BEGIN
;  2566          4073  3         IF .KEY NEQ 0
;  2567          4074  3         THEN
```

```
;  2568          4075  4                         BEGIN
;  2569          4076  4                         !+
;  2570          4077  4                         !  Pass CONTROL Z back to user program via KEY,
;  2571          4078  4                         !  if requested
;  2572          4079  4                         !-
;  2573          4080  4                         CZ_PTR = CZ ;                   ! CZ is literal %X'1A'
;  2574          4081  4                         TERM_PTR = CZ_PTR ;
;  2575          4082  4                         CH$MOVE ( 1, .TERM_PTR, .KEY [DSC$A_POINTER] ) ;
;  2576          4083  3                         END ;
;  2577          4084  2                     END ;
;  2578          4085  1             END ;                                  ! End routine COB$$CONTROL_Z



                                                 0000 00000 COB$$CONTROL_Z:
                                                            .WORD   Save nothing                            ;  4019
                                      5E                    SUBL2   #4, SP
                                            05      AC 95 00005             TSTB    UNIT+1                  ;  4068
                                                    0E 12 00008             BNEQ    1$
                                   00000000G        8F DD 0000A             PUSHL   #COB$_EOFON_ACC         ;  4070
                         00000000G  00              01 FB 00010             CALLS   #1, LIB$STOP
                                                    04 00017             1$:RET
                                      50    08      AC D0 00018             MOVL    KEY, R0                 ;  4073
                                                    0A 13 0001C             BEQL    2$
                                      6E            1A D0 0001E             MOVL    #26, CZ_PTR             ;  4080
                                      51            6E 9E 00021             MOVAB   CZ_PTR, TERM_PTR        ;  4081
                                04    B0            61 90 00024             MOVB    (TERM_PTR), 4(R0)       ;  4082
                                                    04 00028             2$:RET                             ;  4085

; Routine Size:  41 bytes,    Routine Base:  _COB$CODE + 0EB6
```

```
;  2580          4086  1 %SBTTL 'COB$$PARTIAL_SEQ - Partial Escape Sequence'
;  2581          4087  1 ROUTINE COB$$PARTIAL_SEQ ( PARAMETERS : REF VECTOR,
;  2582          4088  1                           UNIT       : VECTOR [2,BYTE] )  :  NOVALUE =
;  2583          4089  1 !++
;  2584          4090  1 ! FUNCTIONAL DESCRIPTION:
;  2585          4091  1 !
;  2586          4092  1 !       The entire Escape sequence did not fit in the initial $GET's buffer.
;  2587          4093  1 !       Perform 1 character Reads until the full sequence is in PUT_HERE or
;  2588          4094  1 !       NEXT_CHAR.
;  2589          4095  1 !
;  2590          4096  1 ! FORMAL PARAMETERS:
;  2591          4097  1 !
;  2592          4098  1 !       PARAMETERS.mlu.ra  Contains data for this routine.
;  2593          4099  1 !
;  2594          4100  1 !       UNIT.rbu.va        Array of two unsigned byte integers.
;  2595          4101  1 !                          The first byte is the unit number designating the
;  2596          4102  1 !                          device from which the string is to be read.
;  2597          4103  1 !                          The second byte indicates whether the routine should
;  2598          4104  1 !                          abort or return to the calling program.
;  2599          4105  1 !
;  2600          4106  1 ! IMPLICIT INPUTS:
;  2601          4107  1 !
;  2602          4108  1 !       NONE
;  2603          4109  1 !
;  2604          4110  1 ! IMPLICIT OUTPUTS:
;  2605          4111  1 !
;  2606          4112  1 !       NONE
;  2607          4113  1 !
;  2608          4114  1 ! ROUTINE VALUE:
;  2609          4115  1 !
;  2610          4116  1 !
;  2611          4117  1 ! SIDE EFFECTS:
;  2612          4118  1 !
;  2613          4119  1 !       NONE
;  2614          4120  1 !
;  2615          4121  1 !--
;  2616          4122  2     BEGIN
;  2617          4123  2
;  2618          4124  2     LOCAL
;  2619          4125  2         RAB                 :  REF $RAB_DECL,
;  2620          4126  2         FUNC_VAL_2,                             ! QIO Function Modifiers for
;  2621          4127  2                                                 ! item list of RMS $GET.
;  2622          4128  2         TERM_CHAR           :  BYTE,            ! $GET input buffer.
;  2623          4129  2         PH                  :  REF VECTOR [1100,BYTE],  ! Address of PUT_HERE
;  2624          4130  2         PH_PTR,                                 ! Pointer to PUT_HERE.
;  2625          4131  2         NC_PTR,                                 ! Pointer to NEXT_CHAR.
;  2626          4132  2         END_OF_TERM         :  INITIAL (0) ;    ! =1 whole Seq in buffer.
;  2627          4133  2
;  2628          4134  2         !+
;  2629          4135  2         ! Bind PARAMETERS to other names.
;  2630          4136  2         !-
;  2631          4137  2
;  2632          4138  2         $BIND_PARAMETERS ;
;  2633          4139  2         PH = .PUT_HERE [DSC$A_POINTER] ;
;  2634          4140  2
;  2635          4141  2         !+
;  2636          4142  2         ! PH_PTR and NC_PTR point to next free space in buffer
```

```
; 2637          4143  2          !  PUT_HERE or NEXT_CHAR.
; 2638          4144             !-
; 2639          4145
; 2640          4146             PH_PTR = .CHARS_READ + .TERM_SIZE ;
; 2641          4147             NC_PTR = 1 ;
; 2642          4148
; 2643          4149             !+
; 2644          4150             !  Read one character at a time until the entire escape sequence has
; 2645          4151             !  been read.
; 2646          4152             !-
; 2647          4153
; 2648          4154             WHILE .END_OF_TERM EQL 0 DO
; 2649          4155                 BEGIN                                          ! Begin loop
; 2650          4156
; 2651          4157                 FUNC_VAL_2 = TRM$M_TM_ESCAPE + TRM$M_TM_NOFILTR + TRM$M_TM_TRMNOECHO
; 2652          4158                                                                 + TRM$M_TM_NOECHO ;
; 2653          4159
; 2654          4160                 RAB = .COBS$AL_WRITE_RAB [ .UNIT[0] ] ;
; 2655          4161                 COBS$RMS_GET ( .RAB, .FUNC_VAL_2, 1, TERM_CHAR ) ;
; 2656          4162
; 2657          4163                 !+
; 2658          4164                 !  Deposit sequence character in appropriate buffer.
; 2659          4165                 !-
; 2660          4166
; 2661          4167                 IF .TERM_IN_NEXT EQL 0
; 2662          4168                 THEN
; 2663          4169 &                   BEGIN
; 2664          4170 &                   !+
; 2665          4171 &                   !  This is a workaround for an RMS bug that did not
; 2666          4172 &                   !  make it into the final code freeze for V4.0.
; 2667          4173 &                   !  The next three lines can be pulled when the RMS fix
; 2668          4174 &                   !  is made.  (see NEXT_CHAR below)
; 2669          4175 &                   !-
; 2670          4176 &                   IF .TERM_SIZE EQL 1                      ! Put first character
; 2671          4177 &                   THEN                                     ! of terminator seq
; 2672          4178 &                       PH [.PH_PTR - 1] = %X'1B' ;          ! into PUT_HERE
; 2673          4179 &
; 2674          4180 &                   PH [.PH_PTR] = .TERM_CHAR ;              ! Put character just
; 2675          4181 &                   PH_PTR = .PH_PTR + 1 ;                   ! read in PUT_HERE
; 2676          4182 &                   TERM_IN_NEXT = 0 ;
; 2677          4183 &                   END
; 2678          4184  3              ELSE
; 2679          4185 &                   BEGIN
; 2680          4186 &                   !+
; 2681          4187 &                   !  This is a workaround for an RMS bug that did not
; 2682          4188 &                   !  make it into the final code freeze for V4.0.
; 2683          4189 &                   !  The next three lines can be pulled when the RMS fix
; 2684          4190 &                   !  is made.
; 2685          4191 &                   !-
; 2686          4192 &                   IF .TERM_SIZE EQL 1                      ! Put first character
; 2687          4193 &                   THEN                                     ! of terminator seq
; 2688          4194 &                       NEXT_CHAR [0] = %X'1B' ;             ! into NEXT_CHAR.
; 2689          4195 &
; 2690          4196 &                   NEXT_CHAR [.NC_PTR] = .TERM_CHAR ;       ! Put character just
; 2691          4197 &                   NC_PTR = .NC_PTR + 1 ;                   ! read in NEXT_CHAR
; 2692          4198  3                  END ;
; 2693          4199  3
```

```
;2694     4200  3              TERM_SIZE = .TERM_SIZE + 1 ;                    ! Total Terminator size
;2695     4201  3
;2696     4202  3              !+
;2697     4203  3              ! Ugly - but it's the only way to check for the end of an
;2698     4204  3              ! escape sequence.  All known KEY escape sequences end in
;2699     4205  3              ! one of these characters and none of these characters fall
;2700     4206  3              ! in the middle of an escape sequence.  This will have to be
;2701     4207  3              ! updated if new escape sequences surface.
;2702     4208  3              !-
;2703     4209  3
;2704     4210  4              IF ((.TERM_CHAR GEQ %C'A'  AND  .TERM_CHAR LEQ  %C'M')  OR
;2705     4211  4                  (.TERM_CHAR GEQ %C'P'  AND  .TERM_CHAR LEQ  %C'S')  OR
;2706     4212  4                  (.TERM_CHAR GEQ %C'l'  AND  .TERM_CHAR LEQ  %C'y')  OR
;2707     4213  4                  (.TERM_CHAR EQL %X'7E'))
;2708     4214  3              THEN
;2709     4215  4                  BEGIN
;2710     4216  4                  END_OF_TERM = 1 ;                           ! Signal completion
;2711     4217  4                  !+
;2712     4218  4                  ! Have to get rid of a possible status RMS$_TNS, Terminator
;2713     4219  4                  ! Not Seen.  Assume success if we have reached this point.
;2714     4220  4                  ! It is not advisable to overwrite data in the RAB but there
;2715     4221  4                  ! is not way to avoid it in this case.
;2716     4222  4                  !-
;2717     4223  4                  RAB [RAB$L_STS] = RMS$_SUC ;
;2718     4224  3                  END ;
;2719     4225  3
;2720     4226  2              END ;                                           ! End loop
;2721     4227  1     END ;                                                    ! End COB$$PARTIAL_SEQ


                                 07FC 00000 COB$$PARTIAL_SEQ:
                                            .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10        ;4087
                          5E                04 C2 00002       SUBL2   #4, SP
                                            5A D4 00005       CLRL    END_OF_TERM           ;4122
                          52          04    AC D0 00007       MOVL    PARAMETERS, R2        ;4132
                          57          24    A2 9E 0000B       MOVAB   36(R2), R7
                          53          04    A2 D0 0000F       MOVL    4(R2), PH             ;4139
                58        1C          A2    67 C1 00013       ADDL3   (R7), 28(R2), PH_PTR  ;4146
                          56                01 D0 00018       MOVL    #1, NC_PTR            ;4147
                          54          08    AC 9A 0001B       MOVZBL  UNIT, R4              ;4160
                                            5A D5 0001F 1$:   TSTL    END_OF_TERM           ;4154
                                            7F 12 00021       BNEQ    10$
                          59        5240    8F 3C 00023       MOVZWL  #21056, FUNC_VAL_2    ;4158
                          55 00000000G0044  D0 00028          MOVL    COB$$AL_WRITE_RAB[R4], RAB ;4160
                                            5E DD 00030       PUSHL   SP                    ;4161
                                            01 DD 00032       PUSHL   #1
                                      0220  8F BB 00034       PUSHR   #^M<R5,R9>
                      FDF3          CF    04 FB 00038       CALLS   #4, COB$$RMS_GET
                          50                6E 9A 0003D       MOVZBL  TERM_CHAR, R0         ;4180
                                      30    A2 D5 00040       TSTL    48(R2)                ;4167
                                            13 12 00043       BNEQ    3$
                          01                67 D1 00045       CMPL    (R7), #1              ;4176
                                            05 12 00048       BNEQ    2$
                       FF A843              1B 90 0004A       MOVB    #27, -1(PH_PTR)[PH]   ;4178
```

```
                                8843              50  90 0004F 28:      MOVB    R0, (PH_PTR)+[PH]                       ; 4180
                                     30           A2  D4 00053          CLRL    48(R2)                                  ; 4182
                                                  10  11 00056          BRB     5$                                      ; 4167
                                   01             67  D1 00058 38:      CMPL    (R7), #1                                ; 4192
                                                  04  12 0005B          BNEQ    4$                                      ;
                             0C    A2             1B  90 0005D          MOVB    #27, 12(R2)                             ; 4194
                             0C  A246             50  90 00061 48:      MOVB    R0, 12(R2)[NC_PTR]                      ; 4196
                                                  56  D6 00066          INCL    NC_PTR                                  ; 4197
                                                  67  D6 00068 58:      INCL    (R7)                                    ; 4200
                             41    8F             50  91 0006A          CMPB    R0, #65                                 ; 4210
                                                  06  1F 0006E          BLSSU   6$                                      ;
                             4D    8F             50  91 00070          CMPB    R0, #77                                 ;
                                                  1E  1B 00074          BLEQU   9$                                      ;
                             50    8F             50  91 00076 68:      CMPB    R0, #80                                 ; 4211
                                                  06  1F 0007A          BLSSU   7$                                      ;
                             53    8F             50  91 0007C          CMPB    R0, #83                                 ;
                                                  12  1B 00080          BLEQU   9$                                      ;
                             6C    8F             50  91 00082 78:      CMPB    R0, #108                                ; 4212
                                                  06  1F 00086          BLSSU   8$                                      ;
                             79    8F             50  91 00088          CMPB    R0, #121                                ;
                                                  06  1B 0008C          BLEQU   9$                                      ;
                             7E    8F             50  91 0008E 88:      CMPB    R0, #126                                ; 4213
                                                  8B  12 00092          BNEQ    1$                                      ;
                                   5A             01  D0 00094 98:      MOVL    #1, END_OF_TERM                         ; 4216
                             08    A5 00010001    8F  D0 00097          MOVL    #65537, -8(RAB)                         ; 4223
                                                FF7D  31 0009F          BRW     1$                                      ; 4154
                                                      04 000A2 108:     RET                                             ; 4227

; Routine Size:  163 bytes,    Routine Base:  _COBSCODE + 0EDF
```

```
;  2723          4228  1 %SBTTL 'COBS$DELETE_KEY - Delete Key processing'
;  2724          4229  1 ROUTINE COBS$DELETE_KEY ( PARAMETERS : REF VECTOR,
;  2725          4230  1                           UNIT       : VECTOR [2,BYTE],
;  2726          4231  1                           FLAGS                        ) : NOVALUE =
;  2727          4232  1 !++
;  2728          4233  1 ! FUNCTIONAL DESCRIPTION:
;  2729          4234  1 !
;  2730          4235  1 !       Delete Key processing.
;  2731          4236  1 !
;  2732          4237  1 ! FORMAL PARAMETERS:
;  2733          4238  1 !
;  2734          4239  1 !       PARAMETERS.mlu.ra  Contains data for this routine.
;  2735          4240  1 !
;  2736          4241  1 !       UNIT.rbu.va        Array of two unsigned byte integers.
;  2737          4242  1 !                          The first byte is the unit number designating the
;  2738          4243  1 !                          device from which the string is to be read.
;  2739          4244  1 !                          The second byte indicates whether the routine should
;  2740          4245  1 !                          abort or return to the calling program.
;  2741          4246  1 !
;  2742          4247  1 !       FLAGS.rlu.v        Screen enhancement flag.
;  2743          4248  1 !
;  2744          4249  1 ! IMPLICIT INPUTS:
;  2745          4250  1 !
;  2746          4251  1 !       NONE
;  2747          4252  1 !
;  2748          4253  1 ! IMPLICIT OUTPUTS:
;  2749          4254  1 !
;  2750          4255  1 !       NONE
;  2751          4256  1 !
;  2752          4257  1 ! ROUTINE VALUE:
;  2753          4258  1 !
;  2754          4259  1 !
;  2755          4260  1 ! SIDE EFFECTS:
;  2756          4261  1 !
;  2757          4262  1 !       NONE
;  2758          4263  1 !
;  2759          4264  1 !--
;  2760          4265  2     BEGIN
;  2761          4266  2
;  2762          4267  2         LOCAL
;  2763          4268  2             RAB     :  REF $RAB_DECL,
;  2764          4269  2             DELETE_BUF :  VECTOR [3, BYTE],
;  2765          4270  2             CHARS_OK   :  INITIAL (0),                  ! Characters not deleted
;  2766          4271  2             REST_LEN,                                   ! Length yet to be ACCEPTed
;  2767          4272  2                                                         ! after DELETE KEY was hit
;  2768          4273  2             REST_PTR ;                                  ! Where to put rest of
;  2769          4274  2                                                         ! chars on next $GET
;  2770          4275  2 !+
;  2771          4276  2 !   Delete Key processing.  Delete one character at a time.  Delete previous
;  2772          4277  2 !   character typed by backspacing, writing a space to  delete (overwrite)
;  2773          4278  2 !   character, and backspacing again to put cursor in position to continue
;  2774          4279  2 !   ACCEPTing data.
;  2775          4280  2 !   Hitting the DELETE KEY terminated the previous $GET, therefore perform
;  2776          4281  2 !   another $GET to read the rest of the data expected.
;  2777          4282  2 !
;  2778          4283  2 !   Bind PARAMETERS to other names.
;  2779          4284  2 !-
```

```
        $BIND_PARAMETERS ;

        IF ( .FLAGS AND V_NO_ECHO ) NEQ 0
        THEN
            !+
            !   If characters were not echoed to terminal there is no need
            !   to move cursor.
            !-
            BEGIN
            DELETE_BUF [0] = NULL ;                          ! null
            DELETE_BUF [1] = NULL ;
            DELETE_BUF [2] = NULL ;
            END
        ELSE
            BEGIN
            DELETE_BUF [0] = BS ;                            ! Backspace
            DELETE_BUF [1] = BLANK ;                         ! Space
            DELETE_BUF [2] = BS ;                            ! Backspace
            END     ;

            WHILE .TERM_LOC EQL DEL_KEY DO
                BEGIN                                        ! Begin Delete Loop

                CHARS_READ = (.CHARS_READ - 1) ;    ! Decr # of valid input chars
                CHARS_OK   = .CHARS_READ ;          ! Save for left border check

                IF .CHARS_READ LSS 0
                THEN CHARS_READ = 0 ;

                !+
                !   Calculations for $GET
                !-
                REST_LEN   = .ACC_SIZE - .CHARS_READ ;
                REST_PTR   = .PUT_HERE [DSC$A_POINTER] + .CHARS_READ ;

                !+
                !   Check for too many deletes, do not delete more characters
                !   then were input.  Ring bell to signal attempt to go beyond
                !   left border.
                !-
                IF .CHARS_OK LSS 0
                THEN
                    BEGIN
                    COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
                    END
                ELSE
                    !+
                    !   $PUT to Delete one character.
                    !-
                    COB$$RMS_PUT_BUFFER ( DELETE_BUF [0], 3, .FLAGS ) ;

                !+
                !   Continue to Read input
                !-

                RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
```

```
3                 COB$$RMS_GET ( .RAB, .FUNC_VAL, .REST_LEN, .REST_PTR ) ;
3
3                 !+
3                 !  Reset CHARS_READ - Update # of input chars read.
3                 !  Reset TERM_SIZE and TERM_LOC - New terminator ( Note: this could
3                 !                                      be the DELETE KEY again)
3                 !-
3
3                 CHARS_READ = .CHARS_READ + .RAB [RAB$W_RSZ] ;
3                 TERM_SIZE  = .RAB [COB$$B_STV2_LEN] ;
3                 TERM_LOC   = .RAB [COB$$B_STV0_TERM]; ! Terminator location.
3
3                 !+
3                 ! Check for partial sequence error
3                 !-
3
3                 IF .RAB [RAB$L_STS] EQL RMS$_PES
3                 THEN
3                     COB$$PARTIAL_SEQ ( .PARAMETERS, .UNIT ) ;
3
2                 END ;                                    ! End Delete Loop
2
2     !+
2     ! Did the latest $GET come across a terminator ?
2     ! If so, set flag used by COB$$ILLEGAL_TERM to signal that the terminator
2     ! was encountered in this routine.
2     !-
2
2         IF .TERM_SIZE NEQ 0 OR .RAB [RAB$L_STS] EQL RMS$_EOF
2         THEN
2             TERM_FROM_DEL = 1 ;
2
1         END ;                                            ! End COB$$DELETE_KEY


                                   01FC 00000 COB$$DELETE_KEY:
                                              .WORD   Save R2,R3,R4,R5,R6,R7,R8                 ; 4229
                 5E           04   C2 00002   SUBL2   #4, SP                                    ;
                              58   D4 00005   CLRL    CHARS_OK                                  ; 4265
                 53      04   AC   D0 00007   MOVL    PARAMETERS, R3                            ; 4273
                 55      1C   A3   9E 0000B   MOVAB   28(R3), R5                                ;
    07      0C   AC           09   E1 0000F   BBC     #9, FLAGS, 1$                             ; 4288
                              6E   B4 00014   CLRW    DELETE_BUF                                ; 4295
                         02   AE   94 00016   CLRB    DELETE_BUF+2                              ; 4297
                              09   11 00019   BRB     2$                                        ; 4288
                 6E    2008   8F   B0 0001B 1$: MOVW  #8200, DELETE_BUF                         ; 4301
            02   AE           08   90 00020   MOVB    #8, DELETE_BUF+2                          ; 4303
                 54      08   AC   9A 00024 2$: MOVZBL UNIT, R4                                 ; 4341
      0000007F   8F      28   A3   D1 00028 3$: CMPL  40(R3), #127                              ; 4306
                              6F   12 00030   BNEQ    7$                                        ;
                              65   D7 00032   DECL    (R5)                                      ; 4309
                 58           65   D0 00034   MOVL    (R5), CHARS_OK                            ; 4310
                              02   18 00037   BGEQ    4$                                        ; 4312
                              65   D4 00039   CLRL    (R5)                                      ; 4313
```

```
                      57        18   A3  3C 0003B 4$:    MOVZWL  24(R3), REST_LEN                       ; 4318
                      57             65  C2 0003F        SUBL2   (R5), REST_LEN                         ;
        56       04   A3             65  C1 00042        ADDL3   (R5), 4(R3), REST_PTR                  ; 4319
                                     58  D5 00047        TSTL    CHARS_OK                               ; 4326
                                     0C  18 00049        BGEQ    5$                                     ;
                                0C   AC  DD 0004B        PUSHL   FLAGS                                  ; 4329
                                     02  DD 0004E        PUSHL   #2                                     ;
                 FDD9 CF             02  FB 00050        CALLS   #2, COB$$RMS_PUT_BYTE                  ;
                                     0D  11 00055        BRB     6$                                     ; 4326
                                0C   AC  DD 00057 5$:    PUSHL   FLAGS                                  ; 4335
                                     03  DD 0005A        PUSHL   #3                                     ;
                                08   AE  9F 0005C        PUSHAB  DELETE_BUF                             ;
                 FE5C CF             03  FB 0005F        CALLS   #3, COB$$RMS_PUT_BUFFER                ;
                      52 00000000G0044   D0 00064 6$:    MOVL    COB$$AL_WRITE_RAB[R4], RAB             ; 4341
                                     56  DD 0006C        PUSHL   REST_PTR                               ; 4342
                                     57  DD 0006E        PUSHL   REST_LEN                               ;
                                20   A3  DD 00070        PUSHL   32(R3)                                 ;
                                     52  DD 00073        PUSHL   RAB                                    ;
                 FD13 CF             04  FB 00075        CALLS   #4, COB$$RMS_GET                       ;
                      50        22   A2  3C 0007A        MOVZWL  34(RAB), R0                            ; 4350
                      65             50  C0 0007E        ADDL2   R0, (R5)                               ;
                   24 A3        0E   A2  9A 00081        MOVZBL  14(RAB), 36(R3)                        ; 4351
                   28 A3        0C   A2  9A 00086        MOVZBL  12(RAB), 40(R3)                        ; 4352
             000181C8 8F        08   A2  D1 0008B        CMPL    8(RAB), #98760                         ; 4358
                                     93  12 00093        BNEQ    3$                                     ;
                                08   AC  DD 00095        PUSHL   UNIT                                   ; 4360
                                     53  DD 00098        PUSHL   R3                                     ;
                 FEBE CF             02  FB 0009A        CALLS   #2, COB$$PARTIAL_SEQ                   ;
                                     87  11 0009F        BRB     3$                                     ; 4306
                                24   A3  D5 000A1 7$:    TSTL    36(R3)                                 ; 4370
                                     0A  12 000A4        BNEQ    8$                                     ;
             0001827A 8F        08   A2  D1 000A6        CMPL    8(RAB), #98938                         ;
                                     04  12 000AE        BNEQ    9$                                     ;
                   34 A3             01  D0 000B0 8$:    MOVL    #1, 52(R3)                             ; 4372
                                         04 000B4 9$:    RET                                            ; 4374

; Routine Size:  181 bytes,    Routine Base:  _COB$CODE + 0F82
```

```
;  2871          4375  1 %SBTTL 'COB$$ILLEGAL_TERM - Illegal Terminator'
;  2872          4376  1 ROUTINE COB$$ILLEGAL_TERM ( PARAMETERS : REF VECTOR,
;  2873          4377  1                            UNIT       : VECTOR [2,BYTE],
;  2874          4378  1                            FLAGS,
;  2875          4379  1                            KEY        : REF $STR$DESCRIPTOR )  :  NOVALUE =
;  2876          4380  1 !++
;  2877          4381  1 ! FUNCTIONAL DESCRIPTION:
;  2878          4382  1 !
;  2879          4383  1 !       Terminator from previous $GET was illegal - ring terminal bell to
;  2880          4384  1 !       signal this.  Perform another $GET of length 1 to look for another
;  2881          4385  1 !       terminator.  Verify this new terminator.
;  2882          4386  1 !
;  2883          4387  1 ! FORMAL PARAMETERS:
;  2884          4388  1 !
;  2885          4389  1 !       PARAMETERS.mlu.ra  Contains data for this routine.
;  2886          4390  1 !
;  2887          4391  1 !       UNIT.rbu.va        Array of two unsigned byte integers.
;  2888          4392  1 !                          The first byte is the unit number designating the
;  2889          4393  1 !                          device from which the string is to be read.
;  2890          4394  1 !                          The second byte indicates whether the routine should
;  2891          4395  1 !                          abort or return to the calling program.
;  2892          4396  1 !
;  2893          4397  1 !       FLAGS.rlu.v        Screen enhancement flag.
;  2894          4398  1 !
;  2895          4399  1 !       KEY.wt.ds          Destination of the receiving field of the control key.
;  2896          4400  1 !
;  2897          4401  1 ! IMPLICIT INPUTS:
;  2898          4402  1 !
;  2899          4403  1 !       NONE
;  2900          4404  1 !
;  2901          4405  1 ! IMPLICIT OUTPUTS:
;  2902          4406  1 !
;  2903          4407  1 !       NONE
;  2904          4408  1 !
;  2905          4409  1 ! ROUTINE VALUE:
;  2906          4410  1 !
;  2907          4411  1 !
;  2908          4412  1 ! SIDE EFFECTS:
;  2909          4413  1 !
;  2910          4414  1 !       NONE
;  2911          4415  1 !
;  2912          4416  1 !--
;  2913          4417  2     BEGIN
;  2914          4418  2         LOCAL
;  2915          4419  2             RAB :  REF $RAB_DECL,
;  2916          4420  2             FUNC_VAL_2,                                  ! QIO Function Modifiers
;  2917          4421  2             NO_BELL          :  INITIAL (0),             ! =0 ring bell, =1 don't
;  2918          4422  2             LOOK_FOR_TERM    :  INITIAL (0),             ! =1 buffer full, $GET
;  2919          4423  2                                                          ! only for a terminator
;  2920          4424  2             REST_LEN,                                    ! Length yet to be input
;  2921          4425  2             REST_PTR ;                                   ! Where to put rest of
;  2922          4426  2                                                          ! input data
;  2923          4427  2 !+
;  2924          4428  2 ! Bind PARAMETERS to other names.
;  2925          4429  2 !-
;  2926          4430  2
;  2927          4431  2     $BIND_PARAMETERS ;
```

```
;  2928          4432  2
;  2929          4433  2      !+
;  2930          4434  2      !  Note :  If COB$$DELETE_KEY was called before this routine some
;  2931          4435  2      !  special handling is necessary.
;  2932          4436  2      !         It is possible a previous call to COB$$DELETE_KEY would have
;  2933          4437  2      !         filled the input buffer without coming across a terminator.
;  2934          4438  2      !         When the input buffer is full - look for terminator only.
;  2935          4439  2      !
;  2936          4440  2      !         It is also possible that COB$$DELETE_KEY came across a terminator,
;  2937          4441  2      !         therefore it is only necessary to verify the terminator not
;  2938          4442  2      !         perform another $GET.  This is flagged by TERM_FROM_DEL = 1.
;  2939          4443  2      !-
;  2940          4444  2
;  2941          4445  2          WHILE .LEGAL EQL 0 DO
;  2942          4446  2              BEGIN                                           ! Begin Term Loop
;  2943          4447  2              IF .NO_BELL EQL 0 AND .TERM_FROM_DEL EQL 0      ! NO_BELL is set in
;  2944          4448  2              THEN                                            ! $VERIFY_TERMINATOR
;  2945          4449  2                  !+
;  2946          4450  2                  !  Ring bell to signal illegal terminator.
;  2947          4451  2                  !  Don't ring bell if processing the Delete key, or if the
;  2948          4452  2                  !  terminator has come from COB$$DELETE_KEY (wait for
;  2949          4453  2                  !  $VERIFY_TERMINATOR to check terminator).
;  2950          4454  2                  !-
;  2951          4455  2                  BEGIN
;  2952          4456  2                  COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
;  2953          4457  2                  END
;  2954          4458  2              ELSE                                            ! Reset - next time
;  2955          4459  2                  NO_BELL = 0 ;                               ! ring bell
;  2956          4460  2              !+
;  2957          4461  2              !  Is there still data yet to be input ?
;  2958          4462  2              !-
;  2959          4463  2
;  2960          4464  2              IF .TERM_FROM_DEL EQL 0
;  2961          4465  2              THEN
;  2962          4466  2                  BEGIN                                       ! Begin TERM_FROM_DEL=0
;  2963          4467  3
;  2964          4468  3                  IF .ACC_SIZE GTR .CHARS_READ
;  2965          4469  3                  THEN
;  2966          4470  3                      BEGIN
;  2967          4471  3
;  2968          4472  3                      !+
;  2969          4473  3                      !  Calculations for $GET.
;  2970          4474  3                      !-
;  2971          4475  3                      REST_LEN   = .ACC_SIZE - .CHARS_READ ;
;  2972          4476  3                      REST_PTR   = .PUT_HERE [DSC$A_POINTER] + .CHARS_READ ;
;  2973          4477  3
;  2974          4478  3                      !+
;  2975          4479  3                      !  NEVER do a Read of 0 length, this causes an infinite loop
;  2976          4480  3                      !  of bell ringing.
;  2977          4481  3                      !-
;  2978          4482  3                      IF .REST_LEN EQL 0 THEN REST_LEN = 1 ;
;  2979          4483  3
;  2980          4484  3                      RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
;  2981          4485  3                      COB$$RMS_GET ( .RAB, .FUNC_VAL, .REST_LEN, .REST_PTR ) ;
;  2982          4486  3
;  2983          4487  3                      !+
;  2984          4488  3                      !  Update CHARS_READ, TERM_SIZE and TERM_LOC.
```

```
;  2985          4489  5                    !-
;  2986          4490  5
;  2987          4491  5                    CHARS_READ = .CHARS_READ + .RAB [RAB$W_RSZ] ;
;  2988          4492  5                    TERM_SIZE  = .RAB [COB$$B_STV2_LEN] ;
;  2989          4493  5                    TERM_LOC   = .RAB [COB$$B_STV0_TERM] ;
;  2990          4494  5
;  2991          4495  4                    END ;
;  2992          4496  4
;  2993          4497  4                !+
;  2994          4498  4                ! $GET buffer filled but no terminator seen - TERM_SIZE = 0
;  2995          4499  4                ! Do 1 character reads until you hit a terminator that
;  2996          4500  4                ! you can then attempt to verify.
;  2997          4501  4                ! Also trap an End of File ^Z here and do not perform
;  2998          4502  4                ! another $GET, $VERIFY_TERMINATOR will take care of the ^Z.
;  2999          4503  4                ! .LOOK_FOR_TERM EQL 1 case -> came into this routine with
;  3000          4504  4                ! $GET buffer filled but illegal terminator, therefore
;  3001          4505  4                ! we are looking only for a terminator.
;  3002          4506  4                !-
;  3003          4507  4
;  3004          4508  4                IF .ACC_SIZE EQL .CHARS_READ
;  3005          4509  4                THEN LOOK_FOR_TERM = 1 ;
;  3006          4510  5                WHILE (.TERM_SIZE EQL 0 AND .RAB [RAB$L_STS] NEQ RMS$_EOF)
;  3007          4511  4                      OR (.LOOK_FOR_TERM EQL 1 ) DO
;  3008          4512  4
;  3009          4513  5                    BEGIN                                   ! Begin 1 char $GET
;  3010          4514  5                    REST_PTR   = .PUT_HERE [DSC$A_POINTER] + .CHARS_READ ;
;  3011          4515  5
;  3012          4516  5                    FUNC_VAL_2 = TRM$M_TM_ESCAPE + TRM$M_TM_NOFILTR
;  3013          4517  5                                 + TRM$M_TM_TRMNOECHO + TRM$M_TM_NOECHO ;
;  3014          4518  5
;  3015          4519  5                    RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
;  3016          4520  5                    COB$$RMS_GET ( .RAB, .FUNC_VAL_2, 1, .REST_PTR ) ;
;  3017          4521  5
;  3018          4522  5                    !+
;  3019          4523  5                    ! Set TERM_SIZE and TERM_IN_NEXT before possible
;  3020          4524  5                    ! call to COB$$PARTIAL_SEQ.
;  3021          4525  5                    ! If user attempts to input data other than a
;  3022          4526  5                    ! terminator - error.
;  3023          4527  5                    !-
;  3024          4528  5
;  3025          4529  5                    IF .RAB [RAB$W_RSZ] NEQ 0
;  3026          4530  5                    THEN
;  3027          4531  6                        BEGIN
;  3028          4532  6                        COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;   ! Error.
;  3029          4533  6                        TERM_SIZE = 0 ;
;  3030          4534  6                        END
;  3031          4535  5                    ELSE                                    ! Terminator seen.
;  3032          4536  5                        IF .RAB [RAB$L_STS] EQL RMS$_EOF
;  3033          4537  5                        THEN
;  3034          4538  5                            !+
;  3035          4539  5                            ! NOTE:  When Control Z is typed in as the only
;  3036          4540  5                            ! input to a $GET it is not recorded in
;  3037          4541  5                            ! RAB [COB$$B_STV2_LEN] therefore, pull out of
;  3038          4542  5                            ! loop and let $VERIFY_TERMINATOR handle the ^Z,
;  3039          4543  5                            ! but first you have to load the ^Z in
;  3040          4544  5                            ! RAB [COB$$B_STV2_LEN]  as this is where
;  3041          4545  5                            ! $VERIFY_TERMINATOR looks for it.
```

```
5                               !-
6                               BEGIN
6                               TERM_SIZE = 1 ;
6                               RAB [COB$$B_STVO_TERM] = CZ ;
6                               LOOK_FOR_TERM = 0 ;                 ! Set to get out of loop
6                               END
5                           ELSE
6                               BEGIN
6                               LOOK_FOR_TERM = 0 ;                 ! Set to get out of loop
6                               TERM_SIZE = .RAB [COB$$B_STV2_LEN] ;
6                               TERM_LOC  = .RAB [COB$$B_STVO_TERM] ;
5                               END ;
4                       END ;                                       ! End 1 char $GET
4
4                   !+
4                   ! Check for partial sequence error
4                   !-
4
4                   IF .RAB [RAB$L_STS] EQL RMS$_PES
4                   THEN
4                       COB$$PARTIAL_SEQ ( .PARAMETERS, .UNIT ) ;
4
3                   END ;                                           ! End TERM_FROM_DEL=0
3
3               !+
3               ! Now have a Terminator in PUT_HERE.  Reset flags.  Call
3               ! macro to verify Terminator.
3               !-
3
3               TERM_PTR = .PUT_HERE [DSC$A_POINTER] + .CHARS_READ ;
3               TERM_FROM_DEL = 0 ;
3               TERM_IN_NEXT = 0 ;
3               $VERIFY_TERMINATOR ;
3
2               END ;                                               ! End Term Loop
2
1       END ;                                                       ! End COB$$ILLEGAL_TERM
```

```
                          OFFC 00000 COB$$ILLEGAL_TERM:
                                            .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11   ; 4376
     5E             04  C2 00002            SUBL2   #4, SP
                    5A  7C 00005            CLRQ    LOOK_FOR_TERM                          ; 4417
     52       04    AC  D0 00007            MOVL    PARAMETERS, R2                         ; 4425
     56       1C    A2  9E 0000B            MOVAB   28(R2), R6
     55       24    A2  9E 0000F            MOVAB   36(R2), R5
     57       38    A2  9E 00013            MOVAB   56(R2), R7
                    67  D5 00017 1$:        TSTL    (R7)                                   ; 4445
                    01  13 00019            BEQL    2$
                        04 0001B            RET
                    5B  D5 0001C 2$:        TSTL    NO_BELL                                ; 4447
                    11  12 0001E            BNEQ    3$
              34    A2  D5 00020            TSTL    52(R2)
                    0C  12 00023            BNEQ    3$
```

```
                              0C    AC  DD 00025         PUSHL   FLAGS                                   ; 4456
                                    02  DD 00028         PUSHL   #2                                      ;
                  FD4A   CF         02  FB 0002A         CALLS   #2, COB$$RMS_PUT_BYTE                   ;
                                    02  11 0002F         BRB     4$                                      ; 4447
                                    58  D4 00031  3$:    CLRL    NO_BELL                                 ; 4459
                              34    A2  D5 00033  4$:    TSTL    52(R2)                                  ; 4464
                                    03  13 00036         BEQL    5$                                      ;
                                  00CE  31 00038         BRW     15$                                     ;
  66        18   A2           10    00  ED 0003B  5$:    CMPZV   #0, #16, 24(R2), (R6)                   ; 4468
                                    3D  15 00041         BLEQ    7$                                      ;
                         59   18    A2  3C 00043         MOVZWL  24(R2), REST_LEN                        ; 4475
                         59         66  C2 00047         SUBL2   (R6), REST_LEN                          ;
                 58   04 A2         66  C1 0004A         ADDL3   (R6), 4(R2), REST_PTR                   ; 4476
                                    59  D5 0004F         TSTL    REST_LEN                                ; 4482
                                    03  12 00051         BNEQ    6$                                      ;
                         59         01  D0 00053         MOVL    #1, REST_LEN                            ;
                         50   08    AC  9A 00056  6$:    MOVZBL  UNIT, R0                                ; 4484
                         53 00000000G0040 D0 0005A       MOVL    COB$$AL_WRITE_RAB[R0], RAB              ;
                                    58  DD 00062         PUSHL   REST_PTR                                ; 4485
                                    59  DD 00064         PUSHL   REST_LEN                                ;
                              20    A2  DD 00066         PUSHL   32(R2)                                  ;
                                    53  DD 00069         PUSHL   RAB                                     ;
                  FC68   CF         04  FB 0006B         CALLS   #4, COB$$RMS_GET                        ;
                         50   22    A3  3C 00070         MOVZWL  34(RAB), R0                             ; 4491
                         66         50  C0 00074         ADDL2   R0, (R6)                                ;
                         65   0E    A3  9A 00077         MOVZBL  14(RAB), (R5)                           ; 4492
                    28   A2   0C    A3  9A 0007B         MOVZBL  12(RAB), 40(R2)                         ; 4493
  66        18   A2           10    00  ED 00080  7$:    CMPZV   #0, #16, 24(R2), (R6)                   ; 4508
                                    03  12 00086         BNEQ    8$                                      ;
                         5A         01  D0 00088         MOVL    #1, LOOK_FOR_TERM                       ; 4509
                         54   08    AC  9A 0008B  8$:    MOVZBL  UNIT, R4                                ; 4519
                                    65  D5 0008F  9$:    TSTL    (R5)                                    ; 4510
                                    0A  12 00091         BNEQ    10$                                     ;
              0001827A   8F   08    A3  D1 00093         CMPL    8(RAB), #98938                          ;
                                    05  12 0009B         BNEQ    11$                                     ;
                         01         5A  D1 0009D  10$:   CMPL    LOOK_FOR_TERM, #1                       ; 4511
                                    53  12 000A0         BNEQ    14$                                     ;
                 58   04 A2         66  C1 000A2  11$:   ADDL3   (R6), 4(R2), REST_PTR                   ; 4514
                         6E 5240    8F  3C 000A7         MOVZWL  #21056, FUNC_VAL_2                      ; 4517
                         53 00000000G0044 D0 000AC       MOVL    COB$$AL_WRITE_RAB[R4], RAB              ; 4519
                                    58  DD 000B4         PUSHL   REST_PTR                                ; 4520
                                    01  DD 000B6         PUSHL   #1                                      ;
                              08    AE  DD 000B8         PUSHL   FUNC_VAL_2                              ;
                                    53  DD 000BB         PUSHL   RAB                                     ;
                  FC16   CF         04  FB 000BD         CALLS   #4, COB$$RMS_GET                        ;
                              22    A3  B5 000C2         TSTW    34(RAB)                                 ; 4529
                                    0E  13 000C5         BEQL    12$                                     ;
                              0C    AC  DD 000C7         PUSHL   FLAGS                                   ; 4532
                                    02  DD 000CA         PUSHL   #2                                      ;
                  FCA8   CF         02  FB 000CC         CALLS   #2, COB$$RMS_PUT_BYTE                   ;
                                    65  D4 000D1         CLRL    (R5)                                    ; 4533
                                    BA  11 000D3         BRB     9$                                      ; 4529
                                    5A  D4 000D5  12$:   CLRL    LOOK_FOR_TERM                           ; 4550
              0001827A   8F   08    A3  D1 000D7         CMPL    8(RAB), #98938                          ; 4536
                                    09  12 000DF         BNEQ    13$                                     ;
                         65         01  D0 000E1         MOVL    #1, (R5)                                ; 4548
                    0C   A3         1A  90 000E4         MOVB    #26, 12(RAB)                            ; 4549
```

```
                                        A5  11 000E8        BRB     9$                              ; 4536
                          65       0E   A3  9A 000EA 13$:   MOVZBL  14(RAB), (R5)                   ; 4555
                 28       A2       0C   A3  9A 000EE        MOVZBL  12(RAB), 40(R2)                 ; 4556
                                        9A  11 000F3        BRB     9$                              ; 4510
           000181C8       8F       08   A3  D1 000F5 14$:   CMPL    8(RAB), #98760                  ; 4564
                                        0A  12 000FD        BNEQ    15$                             ;
                                   08   AC  DD 000FF        PUSHL   UNIT                            ; 4566
                                        52  DD 00102        PUSHL   R2                              ;
               FD9F       CF            02  FB 00104        CALLS   #2, COB$$PARTIAL_SEQ            ;
    2C    A2     04       A2            66  C1 00109 15$:   ADDL3   (R6), 4(R2), 44(R2)             ; 4575
                                   30   A2  7C 0010F        CLRQ    48(R2)                          ; 4577
                          01            65  D1 00112        CMPL    (R5), #1                        ;
                                        3B  12 00115        BNEQ    18$                             ;
                 2C       A2       0C   A3  9E 00117        MOVAB   12(RAB), 44(R2)                 ;
                          51       0C   A3  9A 0011C        MOVZBL  12(RAB), R1                     ;
                          09            51  91 00120        CMPB    R1, #9                          ;
                                        05  13 00123        BEQL    16$                             ;
                          0D            51  91 00125        CMPB    R1, #13                         ;
                                        0D  12 00128        BNEQ    17$                             ;
                          50       10   AC  D0 0012A 16$:   MOVL    KEY, R0                         ;
                                        62  13 0012E        BEQL    21$                             ;
                 04       B0       2C   B2  90 00130        MOVB    @44(R2), @4(R0)                 ;
                                        5B  11 00135        BRB     21$                             ;
                          1A            51  91 00137 17$:   CMPB    R1, #26                         ;
                                        24  13 0013A        BEQL    19$                             ;
                 7F       8F            51  91 0013C        CMPB    R1, #127                        ;
                                        55  12 00140        BNEQ    22$                             ;
                          7E       08   AC  7D 00142        MOVQ    UNIT, -(SP)                     ;
                                        52  DD 00146        PUSHL   R2                              ;
               FDFE       CF            03  FB 00148        CALLS   #3, COB$$DELETE_KEY             ;
                          5B            01  D0 0014D        MOVL    #1, NO_BELL                     ;
                                        49  11 00150        BRB     23$                             ;
                                        66  D5 00152 18$:   TSTL    (R6)                            ;
                                        25  12 00154        BNEQ    20$                             ;
           0001827A       8F       08   A3  D1 00156        CMPL    8(RAB), #98938                  ;
                                        1B  12 0015E        BNEQ    20$                             ;
          32     0C       AC            0B  E0 00160 19$:   BBS     #11, FLAGS, 22$                 ;
                                   0C   AC  DD 00165        PUSHL   FLAGS                           ;
                                        52  DD 00168        PUSHL   R2                              ;
              0000V       CF            02  FB 0016A        CALLS   #2, COB$$CLEAN_UP               ;
                                   10   AC  DD 0016F        PUSHL   KEY                             ;
                                   08   AC  DD 00172        PUSHL   UNIT                            ;
               FD05       CF            02  FB 00175        CALLS   #2, COB$$CONTROL_Z              ;
                                            04 0017A        RET                                     ;
                                   10   AC  D5 0017B 20$:   TSTL    KEY                             ;
                                        17  13 0017E        BEQL    22$                             ;
                                   10   AC  DD 00180        PUSHL   KEY                             ;
                                        65  DD 00183        PUSHL   (R5)                            ;
                                   2C   A2  9F 00185        PUSHAB  44(R2)                          ;
          00000000G       00            03  FB 00188        CALLS   #3, COB$$CONTROL_KEY            ;
                          05            50  E9 0018F        BLBC    R0, 22$                         ;
                          67            01  D0 00192 21$:   MOVL    #1, (R7)                        ;
                                        04  11 00195        BRB     23$                             ;
                                        67  D4 00197 22$:   CLRL    (R7)                            ;
                                        65  D4 00199        CLRL    (R5)                            ;
                                      FE79  31 0019B 23$:   BRW     1$                              ; 4445
                                            04 0019E        RET                                     ; 4582
```

; Routine Size: 415 bytes,   Routine Base: _COB$CODE + 1037

```
;3080     4583  1 %SBTTL 'COB$$CLEAN_UP - Clean up for VAX COBOL'
;3081     4584  1 ROUTINE COB$$CLEAN_UP ( PARAMETERS : REF VECTOR,
;3082     4585  1                         FLAGS
;3083     4586  1                       )  :  NOVALUE =
;3084     4587  1 !++
;3085     4588  1 ! FUNCTIONAL DESCRIPTION:
;3086     4589  1 !
;3087     4590  1 !       Perform clean up before returning control to VAX COBOL.
;3088     4591  1 !       Position cursor after FIELD of POTECTED ACCEPT, $PUT to turn
;3089     4592  1 !       attributes off, and determine if ADVANCING is needed.
;3090     4593  1 !
;3091     4594  1 ! FORMAL PARAMETERS:
;3092     4595  1 !
;3093     4596  1 !       PARAMETERS.mlu.ra  Contains data for this routine.
;3094     4597  1 !
;3095     4598  1 !       FLAGS.rlu.v        Screen enhancement flag.
;3096     4599  1 !
;3097     4600  1 ! IMPLICIT INPUTS:
;3098     4601  1 !
;3099     4602  1 !       NONE
;3100     4603  1 !
;3101     4604  1 ! IMPLICIT OUTPUTS:
;3102     4605  1 !
;3103     4606  1 !       NONE
;3104     4607  1 !
;3105     4608  1 ! ROUTINE VALUE:
;3106     4609  1 !
;3107     4610  1 !
;3108     4611  1 ! SIDE EFFECTS:
;3109     4612  1 !
;3110     4613  1 !       NONE
;3111     4614  1 !--
;3112     4615  2     BEGIN
;3113     4616  2         LOCAL
;3114     4617  2             RAB  :  REF $RAB_DECL ;
;3115     4618  2
;3116     4619  2         !+
;3117     4620  2         ! Bind PARAMETERS to other names.
;3118     4621  2         !-
;3119     4622  2
;3120     4623  2         $BIND_PARAMETERS ;
;3121     4624  2
;3122     4625  2         !+
;3123     4626  2         ! Position cursor after FIELD of POTECTED Read.
;3124     4627  2         ! This code is necessary if the # of characters input is less than
;3125     4628  2         ! the # of characters expected.  Move cursor the difference of the
;3126     4629  2         ! two numbers.
;3127     4630  2         ! If DEFAULT has been used move cursor the whole length of the expected
;3128     4631  2         ! size.
;3129     4632  2         !-
;3130     4633  2
;3131     4634  2         IF .YES_PROTECT
;3132     4635  2         THEN
;3133     4636  3             BEGIN
;3134     4637  3                 LOCAL
;3135     4638  3                     MOVE_CURSOR  :  INITIAL (0),          !  Flag
;3136     4639  3                     MOVE_NUM ;                            !  # of positions to
```

```
;  3137          4640  1                                                       ! move cursor
;  3138          4641  1          IF .YES_DEFAULT NEQ 0
;  3139          4642  1          THEN
;  3140          4643  1              BEGIN
;  3141          4644  1              MOVE_NUM = .ACC_SIZE ;                   ! DEFAULT used
;  3142          4645  1              MOVE_CURSOR = 1 ;
;  3143          4646  1              END
;  3144          4647  1          ELSE
;  3145          4648  1              IF .CHARS_READ LSS .ACC_SIZE             ! # of chars input is
;  3146          4649  1              THEN                                     ! less than expected
;  3147          4650  1                  BEGIN
;  3148          4651  1                  MOVE_NUM = .ACC_SIZE - .CHARS_READ ;
;  3149          4652  1                  MOVE_CURSOR = 1 ;
;  3150          4653  1                  END ;
;  3151          4654
;  3152          4655  1          IF .MOVE_CURSOR NEQ 0
;  3153          4656  1          THEN
;  3154          4657  1              BEGIN
;  3155          4658  1                  LOCAL
;  3156          4659  1                      SPACE_BUF : VECTOR [200,BYTE] ;
;  3157          4660  1
;  3158          4661  1              CH$FILL ( BLANK, .MOVE_NUM, SPACE_BUF [0] ) ; ! # of spaces to move
;  3159          4662  1              COB$$RMS_PUT_BUFFER ( SPACE_BUF [0], .MOVE_NUM, .FLAGS ) ;  ! cursor
;  3160          4663  1              END ;
;  3161          4664      END ;
;  3162          4665
;  3163          4666      !+
;                4667      !  $PUT to turn attributes off.
;                4668      !  If no attributes were turned on, there is no need to turn them off.
;                4669      !  OFF_BUF holds escape sequence to turn attributes off.  OFF_LEN holds
;                4670      !  the length of that sequence.
;  3168          4671      !-
;  3169          4672
;  3170          4673      IF .PUT_FLAG NEQ 0
;  3171          4674      THEN
;  3172          4675          COB$$RMS_PUT_BUFFER ( OFF_BUF [0], .OFF_LEN, .FLAGS ) ;
;  3173          4676
;  3174          4677      !+
;                4678      !  Determine if ADVANCING is requested.
;                4679      !  If bit 10 = 0 advancing.  If bit 10 = 1 no advancing.
;                4680      !  Set COB$$AB_PREV[0] - also depending on bit 10, to flag to next COBOL
;                4681      !  statement that advancing/no advancing is required following this
;                4682      !  ACCEPT statement.
;  3180          4683      !-
;  3181          4684
;  3182          4685      IF (.FLAGS AND V_ADV) NEQ 0
;  3183          4686      THEN
;  3184          4687          COB$$AB_PREV[0] = ACC_DNA                        ! Signal Do Not Advance
;  3185          4688      ELSE
;  3186          4689          !+
;                4690          !  Echo carriage return to screen if advancing is called for.
;  3188          4691          !-
;  3189          4692          BEGIN
;  3190          4693          COB$$RMS_PUT_BYTE ( CARR_RET, .FLAGS ) ;
;  3191          4694          COB$$AB_PREV[0] = ACC_ADV ;                      ! Signal ADVance
;  3192          4695          END ;
;  3193          4696  1   END ;                                               ! End of COB$$CLEAN_UP
```

```
                                              01FC 00000 COB$$CLEAN_UP:
                                                                  .WORD    Save R2,R3,R4,R5,R6,R7,R8                   ; 4584
                   58 00000000G   00  9E 00002                    MOVAB    COB$$AB_PREV, R8                            ;
                   5E      FF38   CE  9E 00009                    MOVAB    -200(SP), SP                                ;
                   56        04   AC  D0 0000E                    MOVL     PARAMETERS, R6                              ; 4617
                   38        3C   A6  E9 00012                    BLBC     60(R6), 4$                                  ; 4634
                                  50  D4 00016                    CLRL     MOVE_CURSOR                                 ; 4636
                             40   A6  D5 00018                    TSTL     64(R6)                                      ; 4641
                                  06  13 0001B                    BEQL     1$                                          ;
                   57        18   A6  3C 0001D                    MOVZWL   24(R6), MOVE_NUM                            ; 4644
                                  11  11 00021                    BRB      2$                                          ; 4645
  1C  A6     18  A6       10          00  ED 00023 1$:            CMPZV    #0, #16, 24(R6), 28(R6)                     ; 4648
                                      08  15 0002A                BLEQ     3$                                          ;
                   57        18   A6  3C 0002C                    MOVZWL   24(R6), MOVE_NUM                            ; 4651
                   57        1C   A6  C2 00030                    SUBL2    28(R6), MOVE_NUM                            ;
                   50             01  D0 00034 2$:                MOVL     #1, MOVE_CURSOR                             ; 4652
                                  50  D5 00037 3$:                TSTL     MOVE_CURSOR                                 ; 4655
                                  13  13 00039                    BEQL     4$                                          ;
      57         20       6E          00  2C 0003B                MOVC5    #0, (SP), #32, MOVE_NUM, SPACE_BUF          ; 4661
                                      6E     00040                                                                     ;
                             08   AC  DD 00041                    PUSHL    FLAGS                                       ; 4662
                                  57  DD 00044                    PUSHL    MOVE_NUM                                    ;
                             08   AE  9F 00046                    PUSHAB   SPACE_BUF                                   ;
                 FC1E   CF        03  FB 00049                    CALLS    #3, COB$$RMS_PUT_BUFFER                     ;
                             44   A6  D5 0004E 4$:                TSTL     68(R6)                                      ; 4673
                                  0E  13 00051                    BEQL     5$                                          ;
                             08   AC  DD 00053                    PUSHL    FLAGS                                       ; 4675
                             54   A6  DD 00056                    PUSHL    84(R6)                                      ;
                             48   A6  9F 00059                    PUSHAB   72(R6)                                      ;
                 FC0B   CF        03  FB 0005C                    CALLS    #3, COB$$RMS_PUT_BUFFER                     ;
             04    08   AC        0A  E1 00061 5$:                BBC      #10, FLAGS, 6$                              ; 4685
                        68        05  90 00066                    MOVB     #5, COB$$AB_PREV                            ; 4687
                                      04 00069                    RET                                                  ;
                             08   AC  DD 0006A 6$:                PUSHL    FLAGS                                       ; 4693
                                  7E  D4 0006D                    CLRL     -(SP)                                       ;
                 FB66   CF        02  FB 0006F                    CALLS    #2, COB$$RMS_PUT_BYTE                       ;
                        68        04  90 00074                    MOVB     #4, COB$$AB_PREV                            ; 4694
                                      04 00077                    RET                                                  ; 4696

; Routine Size:  120 bytes,    Routine Base:  _COB$CODE + 11D6
```

```
;  3195          4697  1 %SBTTL 'COB$$RPG_CLEAN_UP - Clean up for VAX RPG'
;  3196          4698  1 ROUTINE COB$$RPG_CLEAN_UP ( FLAGS )  :  NOVALUE =
;  3197          4699  1 !++
;  3198          4700  1 ! FUNCTIONAL DESCRIPTION:
;  3199          4701  1 !
;  3200          4702  1 !       Perform clean up before returning control to VAX RPG.
;  3201          4703  1 !
;  3202          4704  1 ! FORMAL PARAMETERS:
;  3203          4705  1 !
;  3204          4706  1 !       FLAGS.rlu.v     Screen enhancement flag.
;  3205          4707  1 !
;  3206          4708  1 ! IMPLICIT INPUTS:
;  3207          4709  1 !
;  3208          4710  1 !       NONE
;  3209          4711  1 !
;  3210          4712  1 ! IMPLICIT OUTPUTS:
;  3211          4713  1 !
;  3212          4714  1 !       NONE
;  3213          4715  1 !
;  3214          4716  1 ! ROUTINE VALUE:
;  3215          4717  1 !
;  3216          4718  1 !
;  3217          4719  1 ! SIDE EFFECTS:
;  3218          4720  1 !
;  3219          4721  1 !       NONE
;  3220          4722  1 !
;  3221          4723  1 !--
;  3222          4724  2     BEGIN
;  3223          4725  2
;  3224          4726  2     !+
;  3225          4727  2     ! Determine if ADVANCING is requested.
;  3226          4728  2     ! If bit 10 = 0 advancing.  If bit 10 = 1 no advancing.
;  3227          4729  2     ! Set COB$$AB_PREV[0] - also depending on bit 10, to flag to next COBOL
;  3228          4730  2     ! statement that advancing/no advancing is required following this
;  3229          4731  2     ! ACCEPT statement.
;  3230          4732  2     !-
;  3231          4733  2
;  3232          4734  2     IF (.FLAGS AND V_ADV) NEQ 0
;  3233          4735  2     THEN
;  3234          4736  2         COB$$AB_PREV[0] = ACC_DNA                       ! Signal Do Not Advance
;  3235          4737  2     ELSE
;  3236          4738  3         BEGIN
;  3237          4739  3         !+
;  3238          4740  3         ! Echo carriage return to screen if advancing is called for.
;  3239          4741  3         !-
;  3240          4742  3         COB$$RMS_PUT_BYTE ( CARR_RET, .FLAGS ) ;
;  3241          4743  3         COB$$AB_PREV[0] = ACC_ADV ;                     ! Signal ADVance
;  3242          4744  3         END;
;  3243          4745  2
;  3244          4746  1     END ;                                             ! End of COB$$RPG_CLEAN_UP


                                0004 00000 COB$$RPG_CLEAN_UP:
                                                .WORD   Save R2                 ; 4698
```

```
                         52 00000000G   00  9E 00002             MOVAB   COB$$AB_PREV, R2                          ;
                   04    6C             2A  E1 00009             BBC     #42, FLAGS, 1$                          ; 4734
                         62             05  90 0000D             MOVB    #5, COB$$AB_PREV                        ; 4736
                                        04  00010                 RET                                             ;
                               04       AC  DD 00011 1$:         PUSHL   FLAGS                                   ; 4742
                                        7E  D4 00014             CLRL    -(SP)                                   ;
                            FB47   CF   02  FB 00016             CALLS   #2, COB$$RMS_PUT_BYTE                   ;
                         62             04  90 0001B             MOVB    #4, COB$$AB_PREV                        ; 4743
                                        04  0001E                 RET                                             ; 4746

; Routine Size:  31 bytes,    Routine Base:  _COB$CODE + 124E
```

```
;  3246          4747  1 %SBTTL 'COB$$FORMAT_FOUR - Format Four'
;  3247          4748  1 ROUTINE COB$$FORMAT_FOUR  ( UNIT          :  VECTOR [2,BYTE],
;  3248          4749  1                             FLAGS,
;  3249          4750  1                             KEY           :  REF $STR$DESCRIPTOR
;  3250          4751  1                           )  =
;  3251          4752  1
;  3252          4753  1 !++
;  3253          4754  1 ! FUNCTIONAL DESCRIPTION:
;  3254          4755  1 !
;  3255          4756  1 !       This routine handles VAX COBOL ACCEPT Statement FORMAT FOUR,
;  3256          4757  1 !       Control Key.
;  3257          4758  1 !
;  3258          4759  1 ! FORMAL PARAMETERS:
;  3259          4760  1 !
;  3260          4761  1 !       UNIT.rbu.va     Array of two unsigned byte integers.
;  3261          4762  1 !                       The first byte is the unit number designating the
;  3262          4763  1 !                       device from which the string is to be read.
;  3263          4764  1 !                       The second byte indicates whether the routine should
;  3264          4765  1 !                       abort or return to the calling program.
;  3265          4766  1 !                         Byte 2 = 0  -  routine will abort on control z
;  3266          4767  1 !                                        and reprompt on conversion errors.
;  3267          4768  1 !                                = 1  -  ( AT END )
;  3268          4769  1 !                                        routine will return to calling program
;  3269          4770  1 !                                        on control z and reprompt on conversion
;  3270          4771  1 !                                        errors.
;  3271          4772  1 !                                = 2  -  ( ON EXCEPTION )
;  3272          4773  1 !                                        routine will return to calling program
;  3273          4774  1 !                                        on control z and conversion errors.
;  3274          4775  1 !
;  3275          4776  1 !       FLAGS.rlu.v     Screen enhancement flag;
;  3276          4777  1 !
;  3277          4778  1 !       KEY.wt.ds       Destination of the receiving field of the control key.
;  3278          4779  1 !
;  3279          4780  1 ! IMPLICIT INPUTS:
;  3280          4781  1 !
;  3281          4782  1 !       NONE
;  3282          4783  1 !
;  3283          4784  1 ! IMPLICIT OUTPUTS:
;  3284          4785  1 !
;  3285          4786  1 !       NONE
;  3286          4787  1 !
;  3287          4788  1 ! ROUTINE VALUE:
;  3288          4789  1 !
;  3289          4790  1 !
;  3290          4791  1 ! SIDE EFFECTS:
;  3291          4792  1 !
;  3292          4793  1 !       NONE
;  3293          4794  1 !
;  3294          4795  1 !--
;  3295          4796  1
;  3296          4797  2     BEGIN
;  3297          4798  2
;  3298          4799  2     LOCAL
;  3299          4800  2         RAB             :  REF $RAB_DECL,
;  3300          4801  2         FUNC_VAL,                       ! Read QIO Function Modifiers
;  3301          4802  2                                         ! used in item list by RMS
;  3302          4803  2         TERM_PTR,                       ! Pointer to terminator in buffer
```

```
;  3303          4804  2          NEXT_CHAR       : VECTOR [10,BYTE],    ! Buffer to hold terminator sequence
;  3304          4805  2          LEGAL           : INITIAL (0),         ! = 0 if illegal terminator hit
;  3305          4806  2          TMASK           : VECTOR [2] ;         ! Longform terminator mask
;  3306          4807  2
;  3307          4808  2      !+
;  3308          4809  2      ! Terminator mask - EVERY key is treated as a terminator.  Each key pressed
;  3309          4810  2      ! is checked for validity as a terminator.
;  3310          4811  2      ! Valid terminators are Carriage Return, Tab, Control Z, Arrow keys,
;  3311          4812  2      ! PF keys, and the PROFESSIONAL Editing and Top Row Function keys.
;  3312          4813  2      !-
;  3313          4814  2
;  3314          4815  2          TMASK [0] = 32 ;
;  3315          4816  2          TMASK [1] = UPLIT (-1, -1, -1, -1, -1, -1, -1, -1 ) ;
;  3316          4817  2
;  3317          4818  2      !+
;  3318          4819  2      ! Ring the terminal bell if user requests.
;  3319          4820  2      !-
;  3320          4821  2
;  3321          4822  2          IF ( .FLAGS AND V_BELL ) NEQ 0
;  3322          4823  2          THEN
;  3323          4824  2              COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
;  3324          4825  2
;  3325          4826  2      !+
;  3326          4827  2      ! Determine FUNC_VAL - QIO Function Modifiers used by RMS $GET Service.
;  3327          4828  2      ! Set TRM$M_TM_NOECHO to suppress echoing of input characters to the terminal.
;  3328          4829  2      ! Set TRM$M_TM_ESCAPE to allow Escape sequences to act as terminators (Arrow
;  3329          4830  2      ! keys and PF keys and the Professional editing and top row function keys).
;  3330          4831  2      ! Set TRM$M_TM_NOFILTR to allow this routine to handle the DELETE KEY. (not a
;  3331          4832  2      ! valid terminator).
;  3332          4833  2      ! Set TRM$M_TM_TRMNOECHO to suppress echoing of the termination character
;  3333          4834  2      ! (COB$$AB_PREV handles advancing / no advancing).
;  3334          4835  2      !-
;  3335          4836  2
;  3336          4837  2          FUNC_VAL = TRM$M_TM_ESCAPE + TRM$M_TM_NOFILTR + TRM$M_TM_TRMNOECHO
;  3337          4838  2                                                        + TRM$M_TM_NOECHO ;
;  3338          4839  2
;  3339        P 4840  2          $ITMLST_INIT (ITMLST = XAB_ITMLST,                    ! Item list for $GET
;  3340        P 4841  2                        (ITMCOD = TRM$_MODIFIERS,
;  3341        P 4842  2                         BUFSIZ = 0,
;  3342        P 4843  2                         BUFADR = .FUNC_VAL),
;  3343        P 4844  2                        (ITMCOD = TRM$_TERM,
;  3344        P 4845  2                         BUFSIZ = 32,                         ! 32 bytes in TMASK
;  3345          4846  2                         BUFADR = .TMASK[1]) ) ;
;  3346          4847  2      !+
;  3347          4848  2      ! RMS $GET - expect only terminators.  NOTE:  This $GET call is not the
;  3348          4849  2      ! same as the call in routine COB$$RMS_GET.
;  3349          4850  2      !-
;  3350          4851  2
;  3351          4852  2          WHILE .LEGAL EQL 0 DO
;  3352          4853  2              BEGIN                                            ! Begin Loop
;  3353          4854  2
;  3354          4855  2              RAB = .COB$$AL_WRITE_RAB [.UNIT[0]] ;
;  3355          4856  2              RAB [RAB$W_USZ] = 10 ;
;  3356          4857  2              RAB [RAB$L_UBF] = NEXT_CHAR ;
;  3357          4858  2              RAB [RAB$V_ETO] = 1 ;
;  3358          4859  2              RAB [RAB$L_XAB] = XABTRM ;
;  3359          4860  3              WHILE $GET (RAB = .RAB) EQL RMS$_RSA DO $WAIT (RAB = .RAB) ;
```

```
; 3360          4861  3
; 3361          4862  3          IF NOT .RAB [RAB$L_STS]
; 3362          4863  3          THEN
; 3363          4864  3              !+
; 3364          4865  3              ! These are special case status that will be handled later.
; 3365          4866  3              ! (See note below for explanation of missing RMS$_TNS)
; 3366          4867  3              !-
; 3367          4868  4              IF (.RAB [RAB$L_STS] NEQ RMS$_BES AND
; 3368          4869  4              .RAB [RAB$L_STS] NEQ RMS$_EOF AND
; 3369          4870  4              .RAB [RAB$L_STS] NEQ RMS$_PES AND
; 3370          4871  4              .RAB [RAB$L_STS] NEQ RMS$_RTB )
; 3371          4872  3              THEN
; 3372          4873  3                  LIB$STOP (COB$_ERRDURACC, 1, .RAB + RAB$C_BLN, .RAB [RAB$L_STS],
; 3373          4874  3                                                                 .RAB [RAB$L_STV] ) ;
; 3374          4875  2      !+
; 3375          4876  2      !   NOTE:  No need for call to COB$$PARTIAL_SEQ as buffer of 10 bytes
; 3376          4877  2      !          is more than sufficient to hold complete escape sequences.
; 3377          4878  2      !          Most key escape sequences are between 1-4 bytes long.
; 3378          4879  2      !          Status RMS$_TNS, terminator not seen, would signal a need to
; 3379          4880  2      !          call routine COB$$PARTIAL_SEQ.
; 3380          4881  2      !-
; 3381          4882  2          TERM_PTR = NEXT_CHAR[0] ;
; 3382          4883  2
; 3383          4884  2          !+
; 3384          4885  2          ! Check for legal terminator, then copy it to KEY.
; 3385          4886  2          !-
; 3386          4887  2
; 3387          4888  3              IF .RAB [COB$$B_STV2_LEN] EQL 1             ! Terminator is one byte
; 3388          4889  3              THEN
; 3389          4890  4                  BEGIN
; 3390          4891  4                  TERM_PTR = RAB [COB$$B_STV0_TERM] ;
; 3391          4892  4                  SELECTONE .RAB [COB$$B_STV0_TERM] OF
; 3392          4893  4                      SET
; 3393          4894  4                          [ CR,                           ! Carriage Return
; 3394          4895  4                            TAB ] :                       ! Tab
; 3395          4896  4
; 3396          4897  5                              BEGIN
; 3397          4898  5                              CH$MOVE ( 1, .TERM_PTR, .KEY [DSC$A_POINTER] ) ;
; 3398          4899  5                              LEGAL = 1 ;
; 3399          4900  4                              END ;
; 3400          4901  4
; 3401          4902  4                          [OTHERWISE] :                   ! Error - key not a
; 3402          4903  4                                                          ! terminator
; 3403          4904  5                              BEGIN
; 3404          4905  5                              COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
; 3405          4906  5                              LEGAL = 0 ;
; 3406          4907  4                              END ;
; 3407          4908  4                      TES ;
; 3408          4909  4                  END
; 3409          4910  4
; 3410          4911  3              ELSE
; 3411          4912  3                  IF .RAB [RAB$L_STS] EQL RMS$_EOF
; 3412          4913  3                  THEN
; 3413          4914  3                      !+
; 3414          4915  3                      ! CONTROL Z  - the status RMS$_EOF is returned
; 3415          4916  3                      ! from the $Get Service.  ^Z is not stored in
; 3416          4917  3                      ! RAB[RAB$_STV0_TERM].
```

```
;3417     4918  3                    !-
;3418     4919  3
;3419     4920  4                    BEGIN
;3420     4921  4                    IF .UNIT [1]  EQL  0
;3421     4922  4                    THEN
;3422     4923  4                        LIB$STOP ( COB$_EOFON_ACC )      ! Abort
;3423     4924  4                    ELSE
;3424     4925  4                        COB$$CONTROL_Z ( .UNIT, .KEY ) ; ! Return to calling
;3425     4926  4                        RETURN 0 ;                       ! program.
;3426     4927  4                    END
;3427     4928  3                ELSE
;3428     4929  3                    !+
;3429     4930  3                    !  Escape Sequence as Terminator.  COB$$CONTROL_KEY converts
;3430     4931  3                    !  terminator sequences to COBOL defined sequences and fills
;3431     4932  3                    !  in KEY parameter if terminator is legal.
;3432     4933  3                    !-
;3433     4934  3
;3434     4935  4                    BEGIN
;3435     4936  5                    IF NOT ( COB$$CONTROL_KEY (TERM_PTR, .RAB [COB$$B_STV2_LEN],
;3436     4937  5                                                                        .KEY) )
;3437     4938  4                    THEN
;3438     4939  5                        BEGIN                            ! Error, illegal escape
;3439     4940  5                        COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;    ! sequence.
;3440     4941  5                        LEGAL = 0 ;
;3441     4942  5                        END
;3442     4943  4                    ELSE
;3443     4944  4                        LEGAL = 1 ;
;3444     4945  3                    END ;
;3445     4946  2            END ;                                                ! End Loop
;3446     4947  2
;3447     4948  2        !+
;3448     4949  2        !  VAX COBOL Version 1 / Version 3 interaction.
;3449     4950  2        !  Determine if ADVANCING is requested.
;3450     4951  2        !  If bit 10 = 0 advancing.  If bit 10 = 1 no advancing.
;3451     4952  2        !  Set COB$$AB_PREV[0] - also depending on bit 10, to flag to next COBOL
;3452     4953  2        !  statement that advancing/no advancing is required following this
;3453     4954  2        !  ACCEPT statement.
;3454     4955  2        !-
;3455     4956  2
;3456     4957  2        IF (.FLAGS AND V_ADV) NEQ 0
;3457     4958  2        THEN
;3458     4959  2            COB$$AB_PREV[0] = ACC_DNA                    ! Signal- Do Not Advance
;3459     4960  2        ELSE
;3460     4961  2            !+
;3461     4962  2            !  Echo carriage return to screen if advancing is called for.
;3462     4963  2            !-
;3463     4964  3            BEGIN
;3464     4965  3            COB$$RMS_PUT_BYTE ( CARR_RET, .FLAGS ) ;
;3465     4966  3            COB$$AB_PREV[0] = ACC_ADV ;                  ! Signal- ADVance
;3466     4967  3            END ;
;3467     4968  2
;3468     4969  2        RETURN 1 ;
;3469     4970  1        END ;                                        ! End of routine COB$$FORMAT_FOUR


                                    0126D          .BLKB   3
```

```
FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF  01270 P.AAS:  .LONG   -1, -1, -1, -1, -1, -1, -1, -1                ;
                                FFFFFFFF  FFFFFFFF  01288                                                                    ;
                                                                                                                             ;


                                                    01FC 00000 COB$$FORMAT_FOUR:
                                                                       .WORD   Save R2,R3,R4,R5,R6,R7,R8                     ; 4748
                                58 00000000G   00    9E 00002          MOVAB   COB$$AB_PREV, R8                              ;
                                57 00000000G   00    9E 00009          MOVAB   LIB$STOP, R7                                  ;
                                56      FB0C   CF    9E 00010          MOVAB   COB$$RMS_PUT_BYTE, R6                         ;
                                5E             18    C2 00015          SUBL2   #24, SP                                       ;
                                               55    D4 00018          CLRL    LEGAL                                         ; 4797
                          04    AE             20    D0 0001A          MOVL    #32, TMASK                                    ; 4815
                          08    AE       BF    AF    9E 0001E          MOVAB   P.AAS, TMASK+4                                ; 4816
                                54       08    AC    D0 00023          MOVL    FLAGS, R4                                     ; 4822
                    07          54             04    E1 00027          BBC     #4, R4, 1$                                    ;
                                               54    DD 0002B          PUSHL   R4                                            ; 4824
                                               02    DD 0002D          PUSHL   #2                                            ;
                                66             02    FB 0002F          CALLS   #2, COB$$RMS_PUT_BYTE                         ;
                                51       5240  8F    3C 00032 1$:      MOVZWL  #21056, FUNC_VAL                              ; 4838
                                50 00000000'   EF    9E 00037          MOVAB   XAB_ITMLST, $$ITMBLKPTR                       ; 4846
                                               80    D4 0003E          CLRL    ($$ITMBLKPTR)+                                ;
                                80             51    D0 00040          MOVL    FUNC_VAL, ($$ITMBLKPTR)+                      ;
                                               80    D4 00043          CLRL    ($$ITMBLKPTR)+                                ;
                                80 00030020    8F    D0 00045          MOVL    #196640, ($$ITMBLKPTR)+                       ;
                                80       08    AE    D0 0004C          MOVL    TMASK+4, ($$ITMBLKPTR)+                       ;
                                               80    7C 00050          CLRQ    ($$ITMBLKPTR)+                                ;
                                53       04    AC    9A 00052          MOVZBL  UNIT, R3                                      ; 4855
                                               55    D5 00056 2$:      TSTL    LEGAL                                         ; 4852
                                               03    13 00058          BEQL    3$                                            ;
                                            00EB   31    0005A          BRW     14$                                           ;
                                52 00000000G0043   D0 0005D 3$:        MOVL    COB$$AL_WRITE_RAB[R3], RAB                    ; 4855
                          20    A2             0A    B0 00065          MOVW    #10, 32(RAB)                                  ; 4856
                          24    A2       0C    AE    9E 00069          MOVAB   NEXT_CHAR, 36(RAB)                            ; 4857
                          07    A2             10    88 0006E          BISB2   #16, 7(RAB)                                   ; 4858
                          40    A2 00000000'   EF    9E 00072          MOVAB   XABFRM, 64(RAB)                               ; 4859
                                               52    DD 0007A 4$:      PUSHL   RAB                                           ; 4860
                    00000000G   00             01    FB 0007C          CALLS   #1, SYS$GET
                    000182DA    8F             50    D1 00083          CMPL    R0, #99034
                                               0B    12 0008A          BNEQ    5$
                                               52    DD 0008C          PUSHL   RAB
                    00000000G   00             01    FB 0008E          CALLS   #1, SYS$WAIT
                                               E3    11 00095          BRB     4$
                                50       08    A2    D0 00097 5$:      MOVL    8(RAB), R0                                    ; 4862
                                37             50    E8 0009B          BLBS    R0, 6$                                        ;
                    000181C0    8F             50    D1 0009E          CMPL    R0, #98752                                    ; 4868
                                               2E    13 000A5          BEQL    6$                                            ;
                    0001827A    8F             50    D1 000A7          CMPL    R0, #98938                                    ; 4869
                                               25    13 000AE          BEQL    6$                                            ;
                    000181C8    8F             50    D1 000B0          CMPL    R0, #98760                                    ; 4870
                                               1C    13 000B7          BEQL    6$                                            ;
                    000181A8    8F             50    D1 000B9          CMPL    R0, #98728                                    ; 4871
                                               13    13 000C0          BEQL    6$                                            ;
                                         0C    A2    DD 000C2          PUSHL   12(RAB)                                       ; 4874
                                               50    DD 000C5          PUSHL   R0                                            ; 4873
                                         44    A2    9F 000C7          PUSHAB  68(RAB)                                       ;
                                               01    DD 000CA          PUSHL   #1                                            ;
```

```
                        00000000G  8F  DD 000CC          PUSHL   #COB$_ERRDURACC                                          ;
                               67  05  FB 000D2          CALLS   #5, LIB$STOP                                            ;
                               6E  0C  AE  9E 000D5 6$:    MOVAB   NEXT_CHAR, TERM_PTR                                    ; 4882
                               01  0E  A2  91 000D9        CMPB    14(RAB), #1                                             ; 4888
                                       1D  12 000DD        BNEQ    8$                                                      ;
                               6E  0C  A2  9E 000DF        MOVAB   12(RAB), TERM_PTR                                       ; 4891
                               50  0C  A2  9A 000E3        MOVZBL  12(RAB), R0                                             ; 4892
                               09      50  91 000E7        CMPB    R0, #9                                                  ; 4894
                                       05  13 000EA        BEQL    7$                                                      ;
                               0D      50  91 000EC        CMPB    R0, #13                                                 ;
                                       46  12 000EF        BNEQ    11$                                                     ;
                               50  0C  AC  D0 000F1 7$:    MOVL    KEY, R0                                                 ; 4898
                         04    B0  00  BE  90 000F5        MOVB    @TERM_PTR, @4(R0)                                       ;
                                       46  11 000FA        BRB     12$                                                     ; 4899
                   0001827A    8F  08  A2  D1 000FC 8$:    CMPL    8(RAB), #98938                                          ; 4912
                                       1D  12 00104        BNEQ    10$                                                     ;
                                   05  AC  95 00106        TSTB    UNIT+1                                                  ; 4921
                                       0B  12 00109        BNEQ    9$                                                      ;
                        00000000G  8F  DD 0010B            PUSHL   #COB$_EOFON_ACC                                         ; 4923
                               67      01  FB 00111        CALLS   #1, LIB$STOP                                            ;
                                       49  11 00114        BRB     17$                                                     ;
                               0C  AC  DD 00116 9$:        PUSHL   KEY                                                     ; 4925
                               04  AC  DD 00119            PUSHL   UNIT                                                    ;
                         0106  C6      02  FB 0011C        CALLS   #2, COB$$CONTROL_Z                                      ;
                                       3C  11 00121        BRB     17$                                                     ; 4926
                               0C  AC  DD 00123 10$:       PUSHL   KEY                                                     ; 4937
                               7E  0E  A2  9A 00126        MOVZBL  14(RAB), -(SP)                                          ; 4936
                                   08  AE  9F 0012A        PUSHAB  TERM_PTR                                                ;
                   00000000G   00      03  FB 0012D        CALLS   #3, COB$$CONTROL_KEY                                    ;
                               0B      50  E8 00134        BLBS    R0, 12$                                                 ;
                                       54  DD 00137 11$:    PUSHL   R4                                                      ; 4940
                                       02  DD 00139        PUSHL   #2                                                      ;
                                   66      02  FB 0013B    CALLS   #2, COB$$RMS_PUT_BYTE                                   ;
                                       55  D4 0013E        CLRL    LEGAL                                                   ; 4941
                                       03  11 00140        BRB     13$                                                     ; 4936
                                   55      01  D0 00142 12$:    MOVL    #1, LEGAL                                       ; 4944
                                   FF0E    31  00145 13$:    BRW     2$                                                      ; 4852
                         05            54  0A  E1 00148 14$:    BBC     #10, R4, 15$                                    ; 4957
                                       68  05  90 0014C        MOVB    #5, COB$$AB_PREV                                        ; 4959
                                       0A  11 0014F        BRB     16$                                                     ;
                                       54  DD 00151 15$:    PUSHL   R4                                                      ; 4965
                                       7E  D4 00153        CLRL    -(SP)                                                   ;
                                   66      02  FB 00155    CALLS   #2, COB$$RMS_PUT_BYTE                                   ;
                                   68      04  90 00158    MOVB    #4, COB$$AB_PREV                                        ; 4966
                                   50      01  D0 0015B 16$:    MOVL    #1, R0                                          ; 4969
                                           04  00 0015E        RET                                                     ;
                                       50  D4 0015F 17$:    CLRL    R0                                                      ; 4970
                                           04  00 00161        RET                                                     ;

; Routine Size:  354 bytes,    Routine Base:  _COB$CODE + 1290

; 3470          4971  1
; 3471          4972  1     END                                             ! End of module COB$ACCEPT
; 3472          4973  0     ELUDOM
```

```
;                                  PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  _COB$DATA                       88  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _COB$CODE                     5106  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;
;
;                               Library Statistics
;                                          --------- Symbols ---------
;       File                               Total    Loaded   Percent      Pages Mapped    Processing Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1         9776       153         1              581           00:00.7
;  _$255$DUA28:[COBRTL.OBJ]SMGLIB.L32;1       469        10         2               38           00:00.2
;
;
;                                  COMMAND QUALIFIERS
;
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:COBACCEPT/OBJ=OBJ$:COBACCEPT MSRC$:COBACCEPT/UPDATE=(ENH$:COBACCEPT
;       )

; Size:          4745 code + 449 data bytes
; Run Time:        01:09.6
; Elapsed Time:    05:34.6
; Lines/CPU Min:    4290
; Lexemes/CPU-Min: 29151
; Memory Used:  626 pages
; Compilation Complete
```

COBCVTRPQ
LIS

COBCVTDQ
LIS

COBCANCEL
LIS

COBCVTQP
LIS

COBACCTIM
LIS

COBCVTPQ
LIS

COBCVTRFQ
LIS

COBCVTQF
LIS

COBCVTRQP
LIS

COBCALL
LIS

COBCVTFQ
LIS

COBACCEPT
LIS

COBCVTRDQ
LIS

COBCVTQD
LIS